

Universidad Católica de Santa María

Facultad de Ciencias e Ingenierías Físicas y Formales

Escuela profesional de Ingeniería de Sistemas



**DESARROLLO DE UN JUEGO SERIO BASADO EN TEST-DRIVEN
LEARNING PARA LA ENSEÑANZA DE MATRICES Y
TRANSFORMACIONES GEOMÉTRICAS EN CURSOS DE
PROGRAMACIÓN DE LOS PRIMEROS AÑOS DE LA ESCUELA
PROFESIONAL DE INGENIERÍA DE SISTEMAS**

Tesis presentada por el bachiller:

Rojas Gómez, Christian Emilio

para optar el Título Profesional de:

Ingeniero de Sistemas

Especialidad en Ingeniería de Software

Asesora de Tesis:

Mg. Castro Gutierrez, Eveling Gloria

Arequipa- Perú

2021

DICTAMEN DEL BORRADOR DE TESIS

UCSM-ERP

UNIVERSIDAD CATÓLICA DE SANTA MARÍA
INGENIERIA DE SISTEMAS
TITULACIÓN CON TESIS
DICTAMEN APROBACIÓN DE BORRADOR

Arequipa, 30 de Junio del 2021

Dictamen: 003403-C-EPIS-2021

Visto el borrador del expediente 003403, presentado por:

2012223811 - ROJAS GOMEZ CHRISTIAN EMILIO

Titulado:

**DESARROLLO DE UN JUEGO SERIO BASADO EN TEST-DRIVEN LEARNING PARA LA ENSEÑANZA
DE MATRICES Y TRANSFORMACIONES GEOMÉTRICAS EN CURSOS DE PROGRAMACIÓN DE LOS
PRIMEROS AÑOS DE LA ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS**

Nuestro dictamen es:

APROBADO

1568 - ROSAS PAREDES KARINA
DICTAMINADOR



1631 - MONTESINOS MURILLO ANGEL FELIPE
DICTAMINADOR



1910 - CASTRO GUTIERREZ EVELING GLORIA
DICTAMINADOR



PRESENTACIÓN

Sr. director de la Escuela Profesional de Ingeniería de Sistemas

Dr. Guillermo Enrique Calderón Ruiz

Sres. Miembros del Jurado Examinador de Tesis

Mg. Ángel Felipe Montesinos Murillo

Mg. Eveling Gloria Castro Gutierrez

Dra. Karina Rosas Paredes

De conformidad con las disposiciones del Reglamento de Grados y Títulos de la Escuela Profesional de Ingeniería de Sistemas, remito a vuestra consideración el presente trabajo de investigación titulado: “Desarrollo de un Juego Serio basado en Test-Driven Learning para la enseñanza de Matrices y Transformaciones Geométricas en cursos de programación de los primeros años de la escuela profesional de Ingeniería de Sistemas”, el mismo que de ser aprobado me permitirá optar el Título Profesional de Ingeniero de Sistemas.

AGRADECIMIENTOS



Principalmente agradecer a Dios, por darme la fuerza para superar todos los obstáculos y la salud necesaria para alcanzar mis metas propuestas.

Quedo muy agradecido con mis padres, familiares y amigos que desde siempre han apoyado en todos los sentidos posibles a conseguir mis objetivos personales y profesionales.

Quiero, además, hacer un agradecimiento especial a mi asesora de Tesis: Mg. Eveling Castro Gutierrez quien asumió el compromiso y me acompañó en la secuencia de toda la investigación

DEDICATORIA

Dedico este trabajo a mis padres, especialmente a mi madre quien me dio la vida y por ser una de las personas mas importantes en mi vida. Dedico espiritualmente este logro a Dios, a todas aquellas personas que ya no se encuentran junto a mi, quienes en vida confiaron en mi y me dieron fuerza a superar todos los desafíos y obstáculos que se presentaron a lo largo de esta etapa. Dedico este esfuerzo también todos mis amigos, familiares y mentores que me apoyaron y motivaron en el transcurso de mi formación profesional como Ingeniero de Sistemas.



RESUMEN

Este trabajo nace de nuestra preocupación por la dificultad en el aprendizaje de cursos de programación y matemáticas en los estudiantes de primeros años de la carrera de Ingeniería de Sistemas. Actualmente en la currícula de este programa en la Universidad Católica de Santa María, los cursos de programación y/o matemáticas impartidas durante los primeros años pueden llegar a ser muy complejos, ocasionando dificultades en su aprendizaje. La solución que plantea el presente trabajo está basada en la construcción de un Juego Serio utilizando el enfoque: Test Driven Learning con el uso de estrategias enseñanza-aprendizaje para promover el aprendizaje de matrices y implementación de operaciones básicas, aplicadas en transformaciones geométricas.

Fue aplicado en cursos de 1er año y de 2do año en los años 2017 y 2018 obteniendo resultados relevantes de la experimentación los cuales se analizaron estadísticamente, utilizando las pruebas de Kolmogorov-Smirnov y Wilcoxon para determinar si existen diferencias significativas, además se realizó una medición de patrones en actividad cerebral utilizando un dispositivo para este fin (Emotiv Insight), finalmente se incorporó encuestas estructuradas a los estudiantes para comprender sus percepciones de la metodología propuesta en relación a la tradicional. Se concluyó que los estudiantes incrementan su motivación al experimentar una metodología basada en juegos, así mismo quedó demostrado que los estudiantes mejoraron sus calificaciones utilizando el videojuego en parejas, se determinó que no existe una diferencia significativa con una metodología tradicional en cuanto conocimientos, pero se puede decir que la metodología experimental generó mayor motivación en los estudiantes.

Palabras clave: TDD, TDL, Estrategias de enseñanza, Transformaciones Geométricas, Deserción, Ingenierías, Juegos Serios.

ABSTRACT

The present study arised from our concern for the difficulty in learning programming and mathematics courses in students of the first years of the Systems Engineering career. Currently in the curriculum of the program of the "Universidad Católica de Santa María", the programming and mathematics courses taught during the first years can become very difficult, causing challenges in their learning. The solution proposed in this study is based on the construction of a Serious Game using the approach: Test-Driven Learning mixed with teaching-learning strategies to promote the learning in Matrices and their basic operations applied in Geometric Transformations.

It was applied in 1st year and 2nd year courses in 2017 and 2018 obtaining relevant results in the experimentation which were statistically analyzed, using the Kolmogorov-Smirnov and Wilcoxon tests to compare if there were significant differences, in addition a measurement was performed of patterns in brain activity using the device Emotiv Insight, finally structured surveys were incorporated to the students to understand their perceptions of the our methodology in relation to the traditional one. It was concluded students increase their motivation by experience using a methodology based on games, it was also found students improved their ratings using the game in pairs, we were determined that there is no significant difference between the traditional methodology and our proposal about the knowledge, but we can say the experimental methodology generated more motivation in the students than a traditional class.

Keywords: TDD, TDL, teaching-learning strategies, Geometric Transformations, Engineering Attrition, Serious Games.

INTRODUCCIÓN

Durante la etapa de formación profesional en la carrera de Ingeniería de Sistemas, varios de los cursos de programación y matemática que son dictados en estos primeros años, resultan ser muy complejos, lo cual es natural debido que para la mayoría son conocimientos totalmente nuevos [1]. Generalmente en cursos de programación, esta dificultad puede o no afectar a los estudiantes, ya que algunos de ellos pueden destacar en estos cursos sin ninguna dificultad, algunos destacan solamente en cursos de matemática, algunos solo en programación y otros en ninguno, esto normalmente refleja la formación anterior en la etapa escolar y la facilidad de aprendizaje de estos estudiantes [2]. Esta diferencia en el aprendizaje conlleva a un estado de frustración para aquellos estudiantes que les resulta un desafío el aprender estos nuevos contenidos, para estos estudiantes es importante aplicar una metodología de enseñanza más efectiva que ayude en su aprendizaje [3]. De lo contrario mantendrán un estado de frustración que finalmente ocasiona una desmotivación en el aprendizaje de estos cursos, en el peor de los casos la deserción de estos [4], [5]. Esta decisión afecta directamente el transcurso del desarrollo de la carrera ocasionando retrasos en la culminación de esta y aún mucho peor, la deserción definitiva en la carrera [6]. Tópico el cual definiremos como el problema principal de esta investigación.

Una metodología de enseñanza implica varios aspectos emocionales y de forma, que pueden afectar directamente al aprendizaje del estudiante. Una mala aplicación de esta es responsable directa de los resultados mencionados anteriormente, es por ello por lo que se considerará “La metodología de enseñanza” como causa principal del problema.

Los cursos de programación son fundamentales en la carrera, para posteriormente poder llevar cursos más especializados en los próximos años que definirán una línea de carrera. Así también los cursos de matemática, resulta que los estudiantes, incluso de años superiores no solo

muestran dificultad en el aprendizaje como se mencionó anteriormente, sino que también no pueden encontrar una relación y una aplicación entre estos cursos, matemática y programación, lo cual provoca que el estudiante considere a la programación como un tema totalmente ajeno a la matemática. Surge la necesidad de introducir estos conceptos a los estudiantes de forma más interactiva, concreta y que relacione los cursos de matemáticas como la base de la programación, en otras palabras, enfocar a los cursos de matemática como la teoría que será aplicada en cursos de programación.

La solución que plantea el presente trabajo de investigación es desarrollar una metodología alternativa la cual está basada en el uso de estrategias de enseñanza-aprendizaje inmersas en un videojuego definido técnicamente como DGBL (Digital Game Based Learning) el cual ha sido diseñado para medir la efectividad de estas estrategias. Así mismo esta metodología de enseñanza utilizada se basa en TDL (Test Driven Learning) la cual será utilizada para promover el aprendizaje en operaciones con matrices, sus aplicaciones en transformaciones geométricas y programación, tópicos abstractos que representan dificultad para los actores en una metodología tradicional.

Fue aplicado en cursos de Pensamiento Computacional (1er Año en los años 2017 y 2018) y Programación II (2do Año en el 2017). Se realizó una evaluación a nivel de conocimientos utilizando un Pre-test y un Post-test del tema propuesto, así mismo también se realizó una evaluación cognitiva utilizando el dispositivo Emotiv Insight que permite el monitoreo en los cambios de la actividad cerebral de los estudiantes, de tal forma que se pueda identificar el nivel de compromiso e interés así como también el nivel de estrés en consecuencia de un aprendizaje significativo, esto permite corroborar los resultados de la prueba de conocimientos (Pre-Test y Post-Test). El análisis de los resultados obtenidos de las pruebas de conocimientos se logra con la aplicación de la prueba de Kolmogorov-Smirnov para determinar la normalidad de los datos. Posteriormente se aplica la prueba de Wilcoxon para datos no paramétricos, con el objetivo de

comprobar si existe una diferencia significativa en las muestras. Estas pruebas establecen que no existe diferencia significativa en las muestras, tanto para los estudiantes que utilizaron la metodología tradicional como de los que usaron la metodología alterna, esto indica que se logró el objetivo de introducir los mismos conocimientos utilizando cualquiera de las metodologías planteadas. Pero quedó demostrado que la metodología alternativa fue más motivadora para los estudiantes.



ÍNDICE DE CONTENIDOS

DICTAMEN DEL BORRADOR DE TESIS

PRESENTACIÓN

AGRADECIMIENTOS

DEDICATORIA

RESUMEN

ABSTRACT

INTRODUCCIÓN

CAPÍTULO I. PLANTEAMIENTO DE LA INVESTIGACIÓN.....	1
1. Planteamiento del Problema.....	2
2. Objetivos de la investigación	3
2.1. General	3
2.2. Específicos	3
3. Preguntas de Investigación.....	4
4. Línea y Sub línea de Investigación a la que corresponde el Problema	4
5. Palabras Clave	4
6. Solución propuesta	4
7. Justificación e importancia.....	4
8. Descripción de la Solución.....	6
CAPÍTULO II. FUNDAMENTOS TEÓRICOS.....	7
1. Estado del arte	8
2. Bases Teóricas del proyecto.....	10

2.1.	Edutainment	10
2.2.	Juegos Serios	11
2.3.	Test Driven Development (TDD)	16
2.4.	Test Driven Learning (TDL)	18
2.5.	Estrategias de Enseñanza-Aprendizaje.....	21
2.6.	Metodologías Agiles (SCRUM).....	23
CAPÍTULO III. MARCO METODOLÓGICO		26
1.	Alcances y limitaciones.....	27
2.	Aporte.....	27
3.	Tipo o nivel de investigación	28
4.	Población y Muestra.....	28
5.	Métodos, Técnicas e Instrumentos de Recolección de Datos	28
6.	Plan de análisis estadístico de los datos	30
7.	Metodología de desarrollo escogida.....	31
8.	Aspectos Disciplinarios.....	31
8.1.	Matrices y operaciones básicas	32
8.2.	Transformaciones geométricas.....	35
9.	Aspectos pedagógicos	37
9.1.	Pedagogía tradicional	37
9.2.	Pedagogía propuesta.....	38

CAPÍTULO IV. DESARROLLO	41
1. Análisis de software	42
1.1. Análisis de requerimientos	42
1.2. Objetivos del aprendizaje	42
1.3. Requerimientos Funcionales	42
1.4. Requerimientos NO Funcionales	52
2. Diseño de Software	53
2.1. Vista de Escenarios (+1)	53
2.2. Vista de procesos.....	61
2.3. Vista lógica.....	65
2.4. Vista de despliegue (Desarrollo).....	71
2.5. Vista física.....	72
3. Conceptualización (Preproducción)	74
3.1. Nombre e iconografía del juego	74
3.2. Narrativa del juego	74
3.3. Diseño de interfaces y prototipos	76
4. Implementación de software	94
4.1. Tecnologías Utilizadas	94
4.2. Configuración inicial.....	96
4.3. Creación del contenido audiovisual	103
4.4. Desarrollo del Backend	106

4.5.	Desarrollo de componentes de juego	113
4.6.	Construcción de las escenas	137
4.7.	Página web de presentación	143
5.	Pruebas	144
5.1.	Objetivo	144
5.2.	Alcance	144
5.3.	Trazabilidad de las pruebas	145
5.4.	Definición de casos de prueba	145
5.5.	Pruebas en beta	152
5.6.	Retroalimentación	154
5.7.	Pruebas Finales	156
CAPÍTULO V. RESULTADOS Y DISCUSIÓN.....		158
1.	Análisis de los resultados	159
2.	Discusión de los resultados	164
CONCLUSIONES		168
TRABAJOS FUTUROS.....		169
REFERENCIAS.....		170
ANEXOS.....		178

ÍNDICE DE FIGURAS

Figura 2.1 Clasificación de Juegos Serios.....	11
Figura 2.2 Clasificación de Juegos Serios según criterios de Conocimiento y Usabilidad	13
Figura 2.3 Aporte de Conocimiento, Intención de Uso y Respuesta	13
Figura 2.4 Proceso Test Driven Development	17
Figura 2.5 Proceso Test Driven Learning	20
Figura 3.1 Emotiv Insight Headset.....	30
Figura 3.2 Representación de matrices en Programación	33
Figura 3.3 Algoritmo de suma de matrices	34
Figura 3.4 Multiplicación de dos matrices	34
Figura 3.5 Algoritmo para multiplicar dos matrices	35
Figura 3.6 Representación de imágenes por compresión de dos dimensiones.....	39
Figura 3.7 Actividad presencial: Representación de imágenes.....	40
Figura 4.1 Vista de casos de uso Jugador Inicio del juego	55
Figura 4.2 Vista de casos de uso Interacción del personaje con los objetos de juego	58
Figura 4.3 Vista de casos de uso para resolución de puzzles de transformaciones y TDL.....	61
Figura 4.4 Diagrama de actividades escenas del nivel Inicial.....	62
Figura 4.5 Diagrama de actividades de escenas del primer Nivel	63
Figura 4.6 Diagrama de actividades de escenas del segundo Nivel.....	64
Figura 4.7 Diagrama de actividades de escenas del tercer Nivel	65
Figura 4.8 Diagrama de clases Matrix Engine	66
Figura 4.9 Diagrama de clases Controlador del personaje.....	67
Figura 4.10 Diagrama de secuencia Resolución puzzle de transformación.....	69
Figura 4.11 Diagrama de secuencia Resolución puzzle TDL	69
Figura 4.12 Diagrama Entidad Relación del Juego Serio	70

Figura 4.13 Diagrama de componentes del Juego Serio	71
Figura 4.14 Diagrama de paquetes del Juego Serio	72
Figura 4.15 Diagrama de despliegue del Juego Serio	73
Figura 4.16 Diseño de icono del Juego Serio	74
Figura 4.17 Diseño del concepto del primer nivel	77
Figura 4.18 Diseño del concepto 2do nivel (Capítulo II).....	78
Figura 4.19 Diseño de concepto 3er nivel (Capítulo III)	78
Figura 4.20 Diseño a mano alzada de las acciones del protagonista.....	79
Figura 4.21 Diseño del Spite-sheet del personaje principal.....	80
Figura 4.22 Diseño a mano alzada de los obstáculos	81
Figura 4.23 Diseño a mano alzada de los paneles de control.....	82
Figura 4.24 Wireframe: Menú principal	84
Figura 4.25 Wireframe: Menú de niveles (Capítulos).....	85
Figura 4.26 Wireframe: Nivel de introducción al juego	86
Figura 4.27 Wireframe: Puzzle de introducción	86
Figura 4.28 Wireframe: Recepción 1er Nivel (Hall).....	87
Figura 4.29 Wireframe: Panel de control del primer obstáculo	88
Figura 4.30 Wireframe: Desactivación del primer obstáculo	88
Figura 4.31 Wireframe: Puzzle de representación de imágenes (Avanzado)	89
Figura 4.32 Wireframe: Puzzle de translación geométrica	90
Figura 4.33 Wireframe: Resolución de una traslación geométrica	90
Figura 4.34 Wireframe: Puzzle de rotación Geométrica.....	91
Figura 4.35 Wireframe: Resolución de una rotación Geométrica.....	92
Figura 4.36 Wireframe: Puzzle de escalamiento geométrico	92
Figura 4.37 Wireframe: Puzzle TDL Casos de prueba para Suma de matrices.....	93

Figura 4.38 Wireframe: Puzzle TDL Implementación Suma de matrices.....	94
Figura 4.39 Instalación de Unity Hub	96
Figura 4.40 Configuración inicial Unity Hub	97
Figura 4.41 Listado de versiones disponibles de Unity	97
Figura 4.42 Listado de versiones disponibles de Unity	98
Figura 4.43 Creación del proyecto para el Juego Serio.....	98
Figura 4.44 Entorno inicial del motor de juegos Unity.....	99
Figura 4.45 Configuración del servidor local.....	100
Figura 4.46 Instalación del Manejador de paquetes PHP :Composer	101
Figura 4.47 Instalación de. Manejador de paquetes PHP:Composer	102
Figura 4.48 Creación del hostname para un entorno desarrollo.....	102
Figura 4.49 Spritesheet: Personaje Principal.....	103
Figura 4.50 Spritesheet: Obstáculos del juego	104
Figura 4.51 Spritesheet: Personaje secundario.....	104
Figura 4.52 Spritesheet: Componentes para los puzzles.....	105
Figura 4.53 Imagen para el fondo principal	105
Figura 4.54 Iconografía del juego	106
Figura 4.55 Migración de la entidad de Usuarios (Users).....	107
Figura 4.56 Migración de la entidad de Capítulos (Chapters)	107
Figura 4.57 Migración de la entidad de Intentos (Attempts)	107
Figura 4.58 Migración de la entidad de Configuraciones (Settings).....	108
Figura 4.59 Esquema final de base de datos (Backend).....	110
Figura 4.60 Habilitación del Usuario para la generación de API Tokens.....	111
Figura 4.61 Método para la autenticación basado en tokens personales	111
Figura 4.62 Protección de rutas desde del controlador de usuarios	112

Figura 4.63 Protección de rutas con excepciones desde el controlador de recursos.....	112
Figura 4.64 Interfaz de la clase Coordenada (Point).....	114
Figura 4.65 Interfaz de la clase Pixel (Box).....	115
Figura 4.66 Interfaz de la clase Matriz (Matrix).....	116
Figura 4.67 Implementación del método Suma de matrices.....	117
Figura 4.68 Implementación del método Multiplicación de matrices.....	117
Figura 4.69 Implementación de la clase Imagen (Image).....	118
Figura 4.70 Referencia poligonal del área de una imagen.....	119
Figura 4.71 Implementación del método pointInTriangle.....	120
Figura 4.72 Implementación del método applyTraslation.....	121
Figura 4.73 Implementación del método applyEscalation.....	122
Figura 4.74 Implementación del método reDrawing.....	123
Figura 4.75 Implementación del método GetBinaryPatterns.....	124
Figura 4.76 Interfaz de la clase Caso de Prueba (TestCase).....	125
Figura 4.77 Implementación del método TestCase.Check.....	126
Figura 4.78 Interfaz de la clase Algorithm (Algorithm).....	126
Figura 4.79 Implementación del método Algorithm.Run.....	127
Figura 4.80 Interfaz de la clase IConnector.....	128
Figura 4.81 Implementación del método IConnector.WaitForRequest.....	129
Figura 4.82 Implementación del método IConnector.Dispatch.....	129
Figura 4.83 Implementación del Controlador de seguridad (Copyright).....	130
Figura 4.84 Ciclo de vida de un controlador de juego (Monobehaviour).....	131
Figura 4.85 Interfaz del controlador del personaje principal.....	132
Figura 4.86 Inicialización del personaje principal.....	133
Figura 4.87 Implementación de la mecánica del jugador principal.....	134

Figura 4.88 Implementación de la física (Gravedad) del personaje principal.....	135
Figura 4.89 Interfaz de la clase PlatformGenerator (Generador de niveles).....	136
Figura 4.90 Método de instanciación aleatoria (Generador de niveles).....	136
Figura 4.91 Prefap del personaje principal.....	137
Figura 4.92 Componentes adjuntos del personaje principal	138
Figura 4.93 Componentes preconstruidos (Prefaps) trampas y obstáculos.....	139
Figura 4.94 Maquina de estado finito de animación del personaje principal.....	140
Figura 4.95 Escenario del nivel introductorio.....	141
Figura 4.96 Escenario del primer nivel	141
Figura 4.97 Escenario del segundo nivel	142
Figura 4.98 Escenario del tercer nivel.....	142
Figura 4.99 Escenario del cuarto nivel.....	142
Figura 4.100 Escenario del Hall o Recepción.....	143
Figura 4.101 Página web de presentación.....	143
Figura 4.102 Aplicación del juego en el grupo 3 de Pensamiento Computacional (2017)....	153
Figura 4.103 Actividad II del puzzle de Representación de imágenes	154
Figura 4.104 Vista audio-descriptiva de una Pista o Hint.....	155
Figura 4.105 Aplicación del Juego Serio en el grupo 4 de Pensamiento Computacional.....	156
Figura 4.106 Aplicación del Juego Serio en el grupo 2 Programación II (2018)	156
Figura 4.107 Clase presencial de matrices en el grupo 4 Programación II (2018)	157
Figura 5.1 Estadísticos descriptivos Generales	159
Figura 5.2 Estadísticos descriptivos entre metodologías	160
Figura 5.3 Rangos entre metodologías	160
Figura 5.4 Test Kruskal-Wallis: No diferencia significativa	160
Figura 5.5 Prueba de correlación entre variables Spearman (post-test y ranking).....	161

Figura 5.6 Prueba de correlación entre variables Spearman (diferencia y ranking)	162
Figura 5.7 Estadísticas de grupo (T-Student).....	162
Figura 5.8 Diagrama de caja y bigotes (Nivel de compromiso/Metodología).....	163
Figura 5.9 Estadísticos descriptivos de percentiles (Caja y bigotes)	164



INDICE DE TABLAS

Tabla 4.1	Objetivos de aprendizaje	42
Tabla 4.2	Historia de usuario 01 – Menú principal.....	43
Tabla 4.3	Historia de usuario 02 – Historia de juego	44
Tabla 4.4	Historia de usuario 03 – Autenticación de usuario	44
Tabla 4.5	Historia de usuario 04 – Niveles de juego.....	44
Tabla 4.6	Historia de usuario 04 – Gameplay	45
Tabla 4.7	Historia de usuario 06 –Abrir una compuerta	45
Tabla 4.8	Historia de usuario 07 – Resolver un puzle de representación de imágenes.....	46
Tabla 4.9	Historia de usuario 08 – Resolver un puzle de transformación geométrica.....	46
Tabla 4.10	Historia de usuario 09 – Resolver un puzle de traslación geométrica	47
Tabla 4.11	Historia de usuario 10 – Resolver un puzle de rotación geométrica.....	47
Tabla 4.12	Historia de usuario 11 – Resolver un puzle de escalamiento geométrica	48
Tabla 4.13	Historia de usuario 12 – Resolver un puzle de trivia	48
Tabla 4.14	Historia de usuario 13 – Ver cronómetro en los puzles	48
Tabla 4.15	Historia de usuario 14 – Obtener puntaje.....	49
Tabla 4.16	Historia de usuario 15 – Recoger pistas (Hints).....	50
Tabla 4.17	Historia de usuario 16 – Leer las pistas (Hints)	50
Tabla 4.18	Historia de usuario 17 – Grabar el progreso de avance.....	50
Tabla 4.19	Historia de usuario 18 – Ver puntaje final	51
Tabla 4.20	Historia de usuario 19 – Recolectar recompensas (Gemas).....	51
Tabla 4.21	Historia de usuario 20 – Denegar el acceso al juego.....	51
Tabla 4.22	Requerimiento No Funcional 01 – Seguridad y control de acceso	52
Tabla 4.23	Requerimiento No Funcional 02 Usabilidad del Juego Serio	52
Tabla 4.24	Requerimiento No Funcional N°3: Multiplataforma (Portabilidad).....	53

Tabla 4.25	Caso de uso N°1: El jugador inicia una nueva partida	54
Tabla 4.26	Caso de uso N°2: El jugador carga una partida anterior.....	54
Tabla 4.27	Caso de uso N°3: El jugador visualiza el ranking general	55
Tabla 4.28	Caso de uso N°4: El jugador carga una partida anterior.....	56
Tabla 4.29	Caso de uso N°5: El jugador recolecta recompensas (Gemas).....	56
Tabla 4.30	Caso de uso N°6: El jugador pierde una vida al colisionar con un obstáculo	57
Tabla 4.31	Caso de uso N°7: El jugador desactiva un obstáculo (Trampa o compuerta)	57
Tabla 4.32	Caso de uso N°8: El jugador aplica una transformación geométrica	58
Tabla 4.33	Caso de uso N°9: El jugador resuelve un algoritmo a través de TDL.....	59
Tabla 4.4.34	Colisión - Efecto del personaje principal	83
Tabla 4.35	Paleta de colores – Personaje Principal.....	103
Tabla 4.36	Lista de controladores: Backend	109
Tabla 4.37	Lista de rutas API: Backend.....	113
Tabla 4.38	Alcance de pruebas: Lista de requerimientos.....	144
Tabla 4.39	Trazabilidad de pruebas	145
Tabla 4.40	Caso de prueba 01 – Iniciar el juego	146
Tabla 4.41	Caso de prueba 02 – Autenticar al usuario	146
Tabla 4.42	Caso de prueba 03 – Validar las acciones básicas	147
Tabla 4.43	Caso de prueba 04 – Realizar un puzle de representación de imágenes	147
Tabla 4.44	Caso de prueba 05 – Realizar un puzle de trivia de matrices.....	148
Tabla 4.45	Caso de prueba 06 – Realizar un puzle de traslación geométrica	149
Tabla 4.46	Caso de prueba 07 – Realizar un puzle de rotación geométrica.....	149
Tabla 4.47	Caso de prueba 08 – Realizar un puzle de escalamiento geométrica.....	150
Tabla 4.48	Caso de prueba 09 – Visualizar el ranking general.....	151
Tabla 4.49	Caso de prueba 10 – Diseñar un algoritmo sobre matrices (TDL).....	151
Tabla 4.50	Plan de ejecución del Juego Serio	152



CAPÍTULO 1.
PLANTEAMIENTO DE LA INVESTIGACIÓN

1. Planteamiento del Problema

En las primeras etapas de la carrera de Ingeniería de Sistemas, según indican las estadísticas proporcionadas por la Oficina de Informática Universidad Católica de Santa María (2015), el incremento en la deserción de la carrera es un problema importante que afecta mayormente a estudiantes de primeros años, según Mori Sánchez (2012) esto puede darse por varias razones: a) Un aprendizaje dificultoso, b) Malas expectativas sobre la carrera, c) La complejidad de los cursos y d) Problemas personales totalmente ajenos a la carrera que sólo competen al estudiante, de los cuales no se puede tener una causa específica. Dentro las causas presentadas “La dificultad en el aprendizaje”, es la mas frecuente y es considerada la característica la más relevante de este campo de investigación, debido a que siempre representa un desafío tanto para los docentes como para los estudiantes.

Durante los primeros años en la carrera se dictan cursos importantes de carácter básico, estos desarrollan la lógica en programación y habilidades matemáticas necesarias para el desarrollo de la carrera en los próximos años. Sin embargo, los estudiantes presentan ciertos desafíos en una etapa, específicamente sobre: a) Tiempo de aprendizaje, b) Relación con conocimientos previos, c) El área de aplicación, d) Desarrollo de buenas practicas y e) evaluación y retroalimentación efectiva [8], [9].

Está claro también que no a todos los estudiantes les resulta complicado el poder desempeñarse correctamente en estas materias, pero para el resto es un desafío en el proceso de aprendizaje y si este no es superado correctamente conlleva finalmente a la frustración del estudiante, ocasionando bajas calificaciones en las primeras fases del curso, finalmente la deserción del curso y en el peor de los casos de la carrera. Es por ello por lo que surge la necesidad de aplicar metodologías alternativas que no solo motive al estudiante, sino que

también sea beneficioso en cuanto a conocimientos en estas materias de esta primera etapa, definida como la más importante en la carrera de Ingeniería de Sistemas.

2. Objetivos de la investigación

2.1. General

Desarrollar un Juego Serio basado en Test-Driven Learning para la enseñanza de Matrices y Transformaciones Geométricas en cursos de programación de los primeros años de la escuela profesional de Ingeniería de Sistemas.

2.2. Específicos

- Analizar el estado del arte de Juegos Serios y Test Driven Learning para la enseñanza de programación en primeros años de la carrera de Ingeniería de Sistemas.
- Diseñar mecánicas, interfaces, estudio de niveles y *storytelling* de un Juego Serio para la enseñanza de Matrices, operaciones de matrices, y transformaciones geométricas.
- Implementar un Juego Serio basado en TDL utilizando el motor de juegos Unity y la metodología ágil SCRUM.
- Evaluar el Juego Serio basado en TDL utilizando un pre-test y un post-test de conocimientos.
- Evaluar la calidad afectiva (concentración, relajación, compromiso, interés, excitación, afinidad y estrés) del Juego Serio basado en TDL, utilizando el dispositivo neurosensorial Emotiv Insight.

3. Preguntas de Investigación

¿La aplicación de juegos serios facilita la adopción del enfoque TDL en los estudiantes?

¿Cómo se puede implementar Test Driven Learning dentro de un Juego Serio?

¿Es posible reducir el tiempo del aprendizaje en programación utilizando TDL en un Juego Serio?

4. Línea y Sub línea de Investigación a la que corresponde el Problema

Línea: Ingeniería de software.

Sub Línea: Testing de software.

5. Palabras Clave

TDD, Test-Driven Learning, Estrategias de enseñanza, Matrices, deserción, Juegos Serios.

6. Solución propuesta

La solución que plantea el presente trabajo de investigación se basa en el análisis, diseño e implementación de un Juego Serio, que integra el propósito del enfoque **Test Driven Learning**, para la enseñanza de **Matrices, Operaciones básicas y Transformaciones Geométricas**, juego que se desarrollará utilizando el motor de juegos Unity bajo la metodología ágil SCRUM.

7. Justificación e importancia

La problemática aborda las evidencias obtenidas del departamento de informática de los estudiantes de la Escuela Profesional de Ingeniería de Sistemas de la Universidad Católica de Santa María. Según estadísticas presentadas en el año 2011, el total de la deserción del alumnado en los 5 años de estudio fue de 52%, lo que muestra una brecha

ingreso-egreso que permite tener sólo un 48% de egresados respecto al 100% de ingresantes. Asimismo, estas estadísticas de deserción en los primeros años indican que la mayor deserción promedio de las promociones del 2012 al 2016, se ubica en el primer año, con un 30%, respecto a un total de 18% que ocurre a partir del 2do año, confirmando así los estudios de Baillie y Fitzgerald (2000), Besterfield-Sacre et al. (1997), Budny & Delaney (2001), Moller-Wong et al., (1997), Sánchez et al. (2012) y Shuman et al. (1999) los cuales determinaron que el mayor índice de deserción se localiza en estos primeros años. Estas cifras son preocupantes en comparación a otras carreras, según estudios de Budny y Delaney (2001) y Sánchez et al. (2012) se debe a la dificultad en el proceso de enseñanza-aprendizaje en los primeros cursos de programación y/o matemáticas.

Es por ello que, para esta situación y por medio de esta investigación, se plantea diseñar e implementar un Juego Serio basado en el enfoque de enseñanza Test-Driven Learning propuesto en Dvornik et al. (2011), D. S. Janzen et al. (2007, 2013), D. S. Janzen and Ryoo (2009) y D. Janzen y Saiedian (2008) con la finalidad mejorar la comprensión y promover el uso de TDD en el desarrollo de las habilidades de programación desde una etapa inicial; y , en base al estudio de Embury et al. (2019) integrar los Juegos Serios y TDL, con la finalidad de afrontar las siguientes limitaciones descritas en Briggs y C. Dudley Girard (2007), Clements y Janzen (2010) y Scatalon et al. (2017): a) La carencia en conocimientos avanzados acerca de testing de software y programación orientada a objetos, b) La incapacidad de los estudiantes para escribir pruebas propias, c) El alcance limitado para desarrollar programas con mayor complejidad debido a la simplicidad de las pruebas y d) la falta de interés de los estudiantes para escribir pruebas antes de la solución. Ofreciendo así, recomendaciones y nuevos desafíos para futuros estudios de juegos serios basados en TDL, donde la docencia de los primeros cursos de programación de las carreras de Tecnología, serán los principales beneficiarios de los resultados.

8. Descripción de la Solución

Dado los desafíos planteados anteriormente, se implementó un Juego Serio que facilite la adopción del enfoque TDL en los estudiantes de los cursos de programación de primeros años. Este producto se construyó en base a un proceso de análisis y observación empírica sobre el contenido académico : Matrices, y Transformaciones geométricas, el cual bajo nuestra perspectiva, debido a su naturaleza orientada a objetos [20] es ideal para llevar a cabo el proyecto. Para ello, se utilizará como base la actividad “Image Representation” del CS Education Research Group (2014) la cual nos permite preparar la introducción al contenido principal (matrices y transformaciones geométricas). El Juego Serio contó con un proceso de pre-producción el cual se constituye del diseño de las mecánicas de juego, interfaces, niveles, historia, diálogos y personajes. Posteriormente se elaboró el Diseño de Software bajo la arquitectura de modelamiento 4+1, el cual será utilizado para la implementación del Juego. Inicialmente, se elaboró y validó la lista de requerimientos, los cuales serán desarrollados de acuerdo con los objetivos de aprendizaje. La dinámica del juego esta alineada los objetivos de TDL [15]: a) Enseñar testing en base a ejemplos, b) Proponer un diseño para un marco de pruebas automatizado y de uso sencillo, y c) Fomentar el uso de TDL a través de la experiencia de juego. Para la validación del proyecto, se diseñó un pre-test y un post-test de conocimientos que permita corroborar el logro del “Propósito” o “Objetivo General” del presente trabajo, a través de un análisis estadístico de los resultados obtenidos. Finalmente, se utilizó el dispositivo de medición neurosensorial Emotiv Insight, el cual ha demostrado su efectividad para la evaluación de Juegos Serios [22]. Dicho dispositivo, se utilizó para realizar una evaluación afectiva (concentración, relajación, compromiso, interés, excitación, afinidad y estrés) tras la experimentación con el Juego Serio propuesto, la cual a su vez será validada con una encuesta de satisfacción aplicada a los estudiantes.



CAPÍTULO 2.

FUNDAMENTOS TEÓRICOS

1. Estado del arte

En la actualidad la problemática de la deserción universitaria es cada vez más analizada, existen estudios como los de Baillie y Fitzgerald (2000), Budny y Delaney (2001), Moller-Wong et al. (1997), Sánchez et al. (2012) y Shuman et al. (1999) asimismo, se han propuesto algunas soluciones para lidiar con este tema como proponen Besterfield-Sacre et al. (1997), Budny y Delaney (2001), Chen et al. (2011), Haungs et al. (2012), Moller-Wong et al. (1997), Sánchez et al. (2012), Wu y Richards (2011) coinciden en determinar que la mayor cantidad de deserción se encuentra en el primer año de universidad, denominado como crítico. Sánchez et al. (2012) define que las causas de la deserción suceden por diversos factores siendo el más común: la dificultad en el proceso de enseñanza-aprendizaje de los contenidos complejos en cursos iniciales (programación y matemáticas). Budny y Delaney (2001) determina que el éxito del primer semestre establece las bases para la finalización de la carrera de ingeniería del estudiante. La estructura de la experiencia educativa y la cultura de la disciplina (actitudes y prácticas de los profesores) tienen un impacto mucho mayor sobre otras causas. Una de las propuesta de la academia para minimizar la deserción universitaria, según Chen et al. (2011) y Wu y Richards (2011) es rediseñar los contenidos de los cursos de los primeros años considerando que los métodos de enseñanza-aprendizaje deben ser cada más activos, y deben permitir garantizar una calidad cognitiva-afectiva en los estudiantes. Estudios como lo establecen Briggs y C. Dudley Girard (2007), Clements y Janzen (2010), Desai et al. (2008), Dvornik et al. (2011), D. Janzen y Saiedian (2008), Scatalon et al. (2020), Stejskal y Siy (2013) y Truscan et al. (2020) proponen una metodología de enseñanza basado en TDD en cursos de programación, el cual consiste, en utilizar el concepto de Test-first (prueba primero) para mejorar la comprensión del estudiante y las habilidades de programación de un estudiante al pensar en el problema (comportamiento) antes de la implementación de la solución

(código), este enfoque se introduce bajo el término de TDL Test-driven learning [14]. El uso de TDD en la enseñanza de programación ha demostrado un incremento en el rendimiento y la calidad de software de los estudiantes [20], [28]. Asimismo, permite a los docentes automatizar la evaluación y calificación del aprendizaje [26], [29]. Sin embargo, dado que este enfoque requiere de conceptos avanzados en Testing de Software y en programación orientada a objetos, entre otras dificultades [9]. En el estudio realizado por D. S. Janzen et al. (2007) los autores concluyeron que es necesario generalizar el enfoque TDL para incrementar la motivación de los estudiantes en una etapa introductoria (primeros años), los cuales presentan resistencia para adoptar esta práctica corroborando los estudios de Desai et al. (2008) y D. Janzen y Saiedian (2008), dichos estudios también revelaron un efecto residual en aquellos estudiantes que iniciaron su aprendizaje bajo el enfoque Test-first, y al realizar evaluación longitudinal del mismo estudio demostró significativamente que los estudiantes que adoptaron una metodología Test-first escribirán más pruebas en otros proyectos que los que adoptaron una metodología de desarrollo tradicional (Test-last).

El estudio presentado por Truscan et al. (2020) diseña un marco de trabajo que integra un mecanismo de calificación automatizado para el enfoque TDL, el cual sirve como una fuente de retroalimentación para los estudiantes en el desarrollo de sus programas, así como también reduce en un 80% el tiempo para la calificación de los proyectos, permitiendo al docente medir el progreso del estudiante de forma automatizada en cualquier momento del curso. Pozsgai Hernani (2014) menciona que para adquirir comprensión conceptual y construir activamente los nuevos conocimientos se debe partir de la experiencia y del aprendizaje previo. Gomez-Alvarez et al. (2016), S. Y. bin S. Hussain et al. (2017), Longstreet y Cooper (2012), Ramirez-Rosales et al. (2016) y Wouters (2017) hacen referencia al estudio del uso de juegos serios en un ámbito educativos y definen a un Juego Serio como aquel juego que tiene una finalidad diferente al entretenimiento. Estudios a lo

largo de los años, han demostrado la efectividad de los juegos para el aprendizaje de temas complejos de ingeniería [35]–[37] y han demostrado su utilidad en la enseñanza en ingeniería de software [35], [38]–[40] específicamente en programación [31]. Asimismo, Aleven et al. (2010) y Carrington et al. (2004) presentan metodologías y procesos de enseñanza para el diseño y análisis de un juego educativo, presenta una serie de características esenciales que este debe cumplir para ser eficaz. Riemer y Schrader (2015) determina que un juego de simulación ofrece un mejor nivel de aprendizaje (conocimiento), así como un juego de aventuras ofrecen un mejor nivel de intención de uso (usabilidad). Estudios recientes han adoptado el uso de aprendizaje basado en juegos para implementar un enfoque basado en pruebas (TDL) demostrando un incremento significativo en la motivación de los estudiantes [17].

2. Bases Teóricas del proyecto

2.1. Edutainment

Este término es utilizado para “encapsular todos los medios educativos de entretenimiento (por ejemplo, películas, juegos, aplicaciones y la red interna) y tiene la intención de ser divertido y educativo” (Kenwright, 2017, p. 2). Estos pueden ser digitales o no digitales, que en su manipulación aportan algún tipo de conocimiento de manera divertida [31]. Entonces se define a este grupo de herramientas como aquellos que además de generar una experiencia lúdica (que resulta altamente placentera al usuario), también permiten arribar conocimiento, creatividad y construcción social al usuario.

2.2. Juegos Serios

Se define Juegos Serios como “Juegos (digitales) utilizados para fines distintos de sólo entretenimiento” [32]. Los cuales se utilizan como medios de comunicación entre el estudiante y conocimiento. [41]

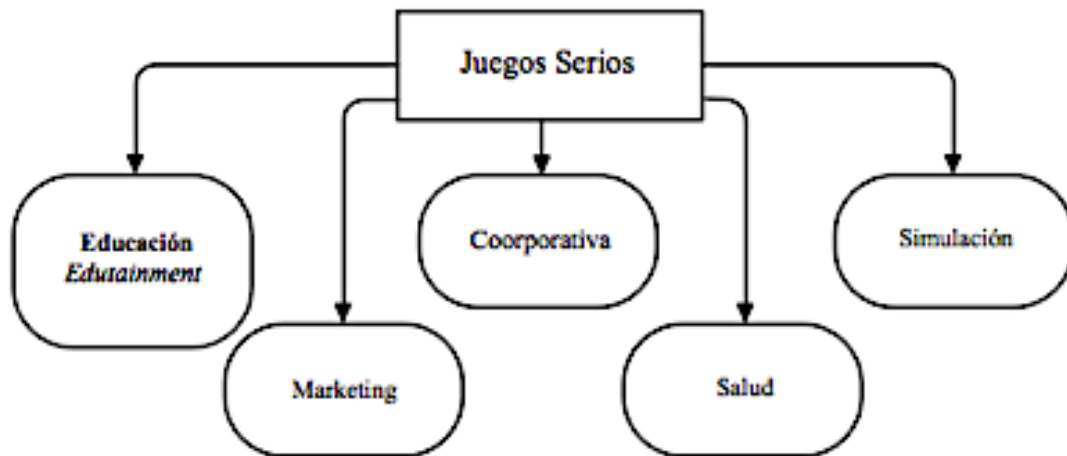


Figura 2.1 Clasificación de Juegos Serios

Fuente: Riemer y Schrader (2015)

Como se muestra en la Figura 2.1, los Juegos Serios son aplicados en varias áreas, pero todos ellos con un objetivo en común, generar conocimiento al usuario [41]. Forma parte de la contribución en el área de Educación del punto anterior (Edutainment).

2.2.1. Características principales

“Un juego educativo tiene que tener éxito en dos objetivos: como una herramienta educativa y como un juego divertido” [39]. De acuerdo con este concepto se define dos características importantes:

- a) **Efectividad de aprendizaje (como herramienta educativa)**

En varios estudios se ha demostrado una eficacia pedagógica en la experimentación con juegos educativos para la enseñanza, presentan una mejora significativa en cuanto a conocimientos [36]. Aunque no se logra una diferencia significativa con la enseñanza tradicional a este nivel, sin embargo el cambio motivacional es notorio [43].

b) **Motivación en el aprendizaje (como un juego divertido)**

Es claro que para llevar un proceso de aprendizaje positivo el estudiante debe estar motivado. La percepción de la calidad afectiva que tiene un estudiante, frente al uso de Juegos Educativos puede provocar efectos positivos (entusiasmo y motivación) como también negativos (nerviosismo y aburrimiento) lo cual determina una intención de uso en la herramienta [41]. Es claro también saber que existen diferencias que en una escala de actitud por género (hombres y mujeres), las mujeres tienen una mayor afectación negativa (aburrimiento y frustración) por lo tanto menos intención de uso, a diferencia de los hombres que muestran una afectación positiva en el uso de videojuegos para el aprendizaje [41].

2.2.2. **Clasificación de Juegos Serios**

Según Riemer y Schrader (2015) clasifica los juegos serios de la siguiente manera:

- a) **Cuestionarios:** Juegos que tienen una respuesta rápida en competencia con otros jugadores y el sistema.
- b) **Simulaciones:** Representaciones del mundo real donde el jugador maneja los eventos Aventuras: Mundos virtuales y una historia donde el estudiante está inmerso en el juego.

- c) **Aventura:** Estos juegos permiten el auto descubrimiento de la información, se caracterizan por darle libertad al jugador de explorar en el entorno de juego. La adquisición de conocimiento se genera a través de la experiencia obtenida.

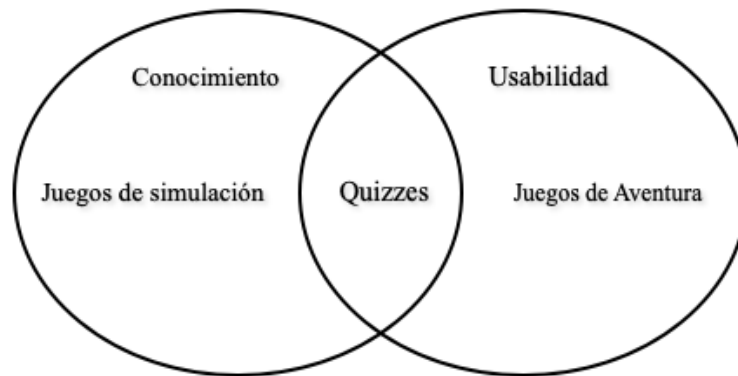


Figura 2.2 Clasificación de Juegos Serios según criterios de Conocimiento y Usabilidad

Fuente: Riemer y Schrader (2015)

Como muestra la Figura 2.2, los juegos de simulación ofrecen mejor nivel de aprendizaje (conocimiento), los juegos de aventura ofrecen un mejor nivel de intención de uso (jugabilidad) y los cuestionarios presentan ambas características.

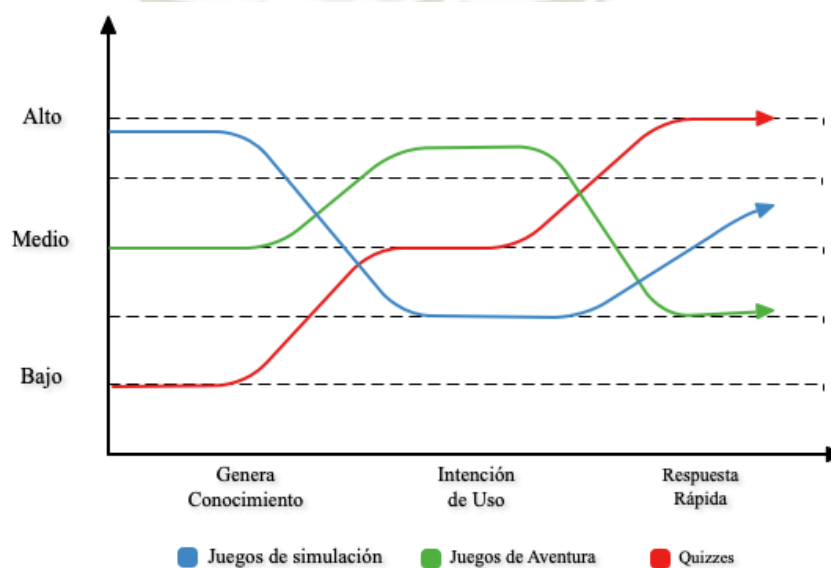


Figura 2.3 Aporte de Conocimiento, Intención de Uso y Respuesta

Fuente: Riemer y Schrader (2015)

Los juegos de simulación generan más conocimiento por experiencia y requiere de conocimientos previos o “Anclaje” [30] para lograr un aprendizaje significativo como se puede observar en la Figura 2.3. Debido a que, un juego de Aventura es más de descubrimiento, no requiere conocimientos previos y genera conocimiento progresivamente, un juego de tipo Cuestionario (Quizzes) requiere de un “Anclaje” mayor debido que este es similar a un examen sólo que más atractivo. Por ejemplo, si se requiere que los usuarios obtengan un mayor conocimiento por experiencia en otras palabras aplicar los conocimientos teóricos para obtener un “Aumento de la retención del aprendizaje” [31] lo ideal es utilizar un Juego de Simulación los cuales tienen mayor popularidad y contribución según Gomez-Alvarez et al. (2016).

Por otro lado, si se desea enseñar progresivamente nuevos contenidos a usuarios inexpertos, lo ideal es utilizar un juego de Aventura que proveen de una mayor intención de uso. Ahora bien, si el objetivo es evaluar el conocimiento y tiempo de respuesta del usuario, lo ideal es aplicar un juego de Cuestionario (Quizzes).

Un aprendizaje basado en juegos en mundos virtuales es una forma de enseñanza centrado en el estudiante que utiliza juegos digitales con fines educativos [38]. Los juegos permiten socializar con otros jugadores, explorar, participar en actividades grupales e individuales para que los estudiantes mejoren sus capacidades de retención [36].

2.2.3. Componentes de los Juegos Serios

Según Alevén et al. (2010) presenta 3 componentes fundamentales que los juegos serios deben contemplar:

- a) **Objetivos de aprendizaje:** Esta parte básicamente trata de especificar detalladamente los objetivos educativos del DGBL a través de los siguientes cuestionamientos.
- **Conocimiento previo:** ¿Qué conocimientos o habilidades deben tener los estudiantes antes de comenzar el juego?
 - **Aprendizaje y retención:** ¿Qué conocimientos o habilidades pueden razonablemente esperar los estudiantes para aprender del juego?
 - **Transferencia potencial:** ¿Qué conocimientos y habilidades podrían aprender de forma directa o indirecta con del juego?
- b) **MDA:** Esta segunda parte se enfoca en el diseño del juego y su composición (Estructura Básica):
- **Mecánica de un juego:** Son todos aquellos componentes con los cuales se elabora un juego (los recursos, reglas, objetivos explícitos, movimientos o acciones y controles disponibles de los jugadores)
 - **Dinámica del juego:** Define el comportamiento resultante tras aplicar la mecánica del juego con la entrada del jugador durante el juego.
 - **Estética de un juego:** Captura la experiencia del jugador, la respuesta emocional o el placer que el juego está diseñado para provocar : sensación, fantasía, narrativa, retos, competencia, descubrimiento y expresión.
- c) **Principios de instrucción:** Principios publicados en diversas investigaciones en ciencias de la educación para su uso didáctico, algunos de estos son:

- **Principio de práctica:** “Los estudiantes reciben mucha y mucha práctica en un contexto donde la práctica no es aburrida” [39]
- **Principio de Goldilocks:** “Las tareas o desafíos no deben ser ni muy fáciles no muy difíciles, este tiene que estar de acuerdo con la habilidad o conocimientos previos del estudiante” [39]
- **Principio de Competencia:** “El estudiante tiene amplia oportunidad de operar dentro, pero al borde de sus recursos, de modo las cosas se sienten como desafiantes, pero no inviables” [39].

Una parte fundamental de un Juego Serio es que sea entretenido y la forma de medir esta sensación es a través del éxito en el logro de los objetivos visuales, donde la mecánica y la dinámica son simplemente herramientas para llegar allí [44].

2.3. Test Driven Development (TDD)

Desarrollo guiado por pruebas de software (TDD) es una práctica popularizada en los últimos años en Ingeniería de Software que se aplica en la fase de desarrollo, esta consta de 4 etapas: a) Escribir las pruebas primero, b) Esperar que las pruebas fallen, c) implementar la solución y d) Refactorizar [20]. Sin embargo, antes de escribir las pruebas, se debe definir la cobertura y alcance de pruebas, esto se logra a través del diseño de casos de prueba, que permite determinar con una mayor precisión la búsqueda de posibles errores reduciendo la cantidad de esfuerzo y tiempo posible por parte del desarrollador [25].

Posterior al diseño, se escriben las pruebas, que generalmente son pruebas unitarias [18]. En una primera instancia, una vez escrita una prueba se debe verificar que esta falle por falta de código que la pueda satisfacer. Una vez que esto ocurre, se procede a escribir el código correspondiente para que la prueba pase satisfactoria, una vez superada esta etapa, es fundamental optimizar el código escrito a lo que se define como “Refactorización”. La

ejecución de las pruebas debe iterarse cada vez que se crea necesario, específicamente después de aplicar alguna refactorización [17].

El objetivo de TDD es lograr un código limpio y funcional [24]. Una forma de establecer y especificar los requerimientos de software es a través de pruebas [25], de este modo, las pruebas garantizarán que el software cumple con los requisitos que se han establecido inicialmente [15].

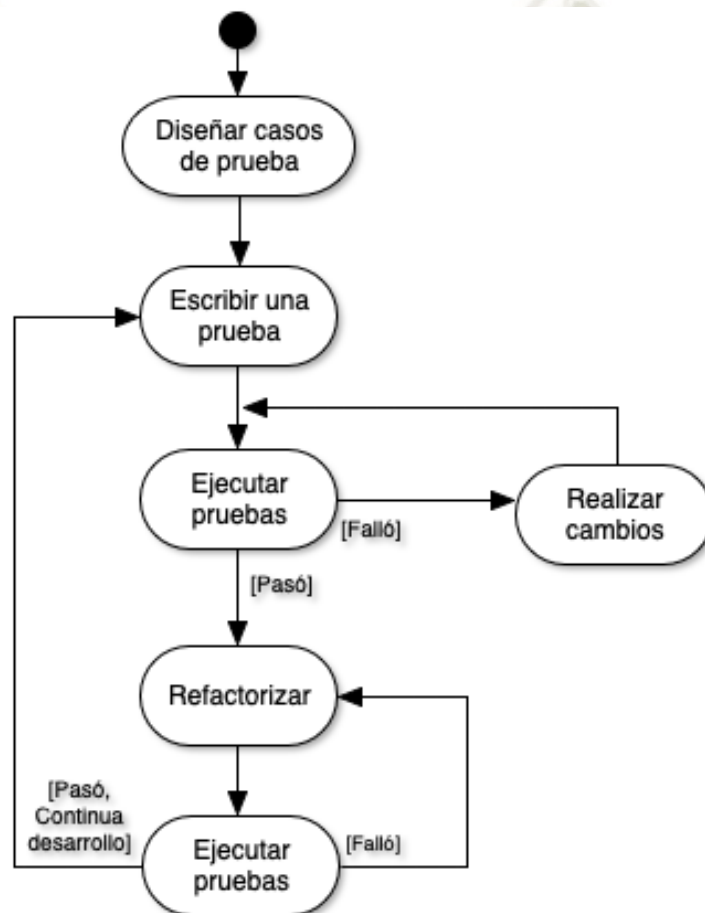


Figura 2.4 Proceso Test Driven Development

Fuente: Elaboración propia (2018)

Como se visualiza en la Figura 2.4, TDD es un proceso iterativo y se compone de las siguientes sub prácticas [16]: a) Automatización: Las pruebas deben ser implementadas en código, esto permite que se puedan ejecutar de manera continua y rápida. Test-first: Escribir la prueba primero antes del código que se debe probar.

Refactorización: Cambiar el diseño, eliminar la redundancia del código, sin cambiar la funcionalidad, con la ejecución de pruebas como respaldo.

2.4. Test Driven Learning (TDL)

Test Driven Learning es un enfoque en la enseñanza de programación y pruebas, está basado en la práctica de ingeniería de software “Test Driven Development” (TDD) por lo tanto proporciona todas las ventajas que este marco metodológico ofrece. Test Driven Learning o TDL es considerada una manera alternativa para la enseñanza de programación propuesto por D. S. Janzen y Ryoo (2009), el cual ha demostrado su efectividad para la enseñanza de programación en primeros años [9], [15], [20]. Este enfoque consiste en realizar las pruebas primero antes de implementar el código de producción [15]. Esto permite que los estudiantes piensen primero en el problema antes que la solución, por lo tanto, promueve que los estudiantes puedan aprender de sus errores [28]. En la actualidad, existen varias herramientas que ayudan a los estudiantes organizar y ejecutar las pruebas con frecuencia, ofreciendo una retroalimentación rápida, lo cual facilita el aprendizaje y desarrolla buenas prácticas en la formación de futuros programadores [26].

El uso de TDL con otras herramientas permite además de visualizar los casos de pruebas, la implementación al mismo tiempo y recibir una retroalimentación rápida al ejecutar los casos de prueba de forma interactiva para el estudiante [13].

El objetivo del enfoque TDL es lograr que el estudiantes piense en el problema primero, posteriormente identifique casos, analice formas de donde podría fallar y evaluar el diseño. Este enfoque está diseñando para alentar a los estudiantes a evaluar la corrección de sus códigos y obliga al estudiante a pensar en posibles errores que puedan ocurrir de tal forma que puedan asegurar que estos errores no se presenten en desarrollo del código [13].

Test Driven Learning o TDL mejora la capacidad de pensamiento de un estudiante respecto al problema, utilizando casos de prueba y en base a ello obtener una solución adecuada.

2.4.1. Proceso del TDL

Test Driven Learning permite que los estudiantes escriban pequeños casos de prueba para entender el problema antes de realizar la implementación. El uso de TDL con otras herramientas permite además de visualizar los casos de pruebas, la implementación al mismo tiempo y recibir una retroalimentación rápida al ejecutar los casos de prueba de forma interactiva para el estudiante [13]. El proceso de integración TDL se define según Stejskal y Siy (2013):

- Familiarizarse con el uso de funciones (este paso es opcional en caso los estudiantes ya conozcan sobre este tema)
- Los casos de prueba iniciales son otorgados por el instructor para que los estudiantes se familiaricen con la sintaxis.
- Los estudiantes escriben nuevos casos de pruebas.
- Integrar las pruebas (Test) y el código al mismo tiempo (Proceso TDD).

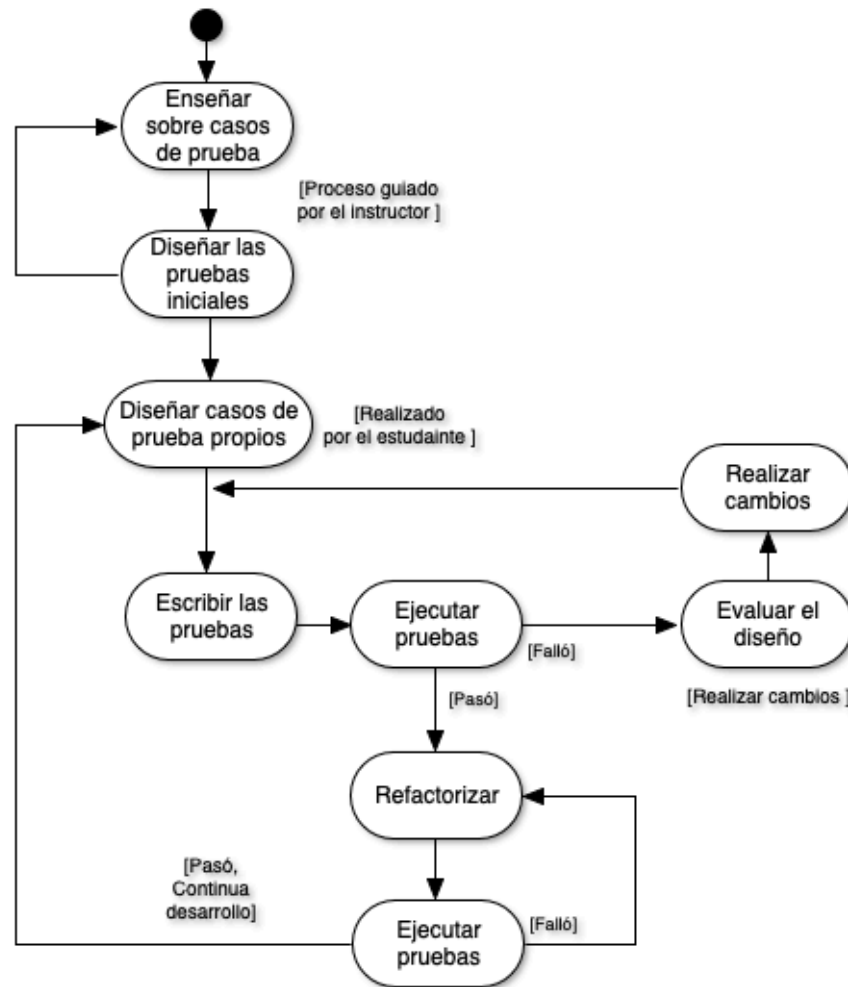


Figura 2.5 Proceso Test Driven Learning

Fuente: Elaboración propia (2018)

Como se puede observar en la Figura 2.5, este proceso ayuda aclarar los requerimientos del problema para el estudiante, haciendo que este piense en el problema primero, posteriormente identifique casos, analice formas de donde podría fallar y evaluar el diseño. Este enfoque está diseñado para alentar a los estudiantes a evaluar la corrección de sus códigos, siendo esta una característica fundamental para el proceso de aprendizaje en base a pruebas [29]. Test Driven Learning o TDL mejora la capacidad de pensamiento de un estudiante respecto al problema, utilizando casos de prueba y en base a ello obtener una solución adecuada [15].

2.5. Estrategias de Enseñanza-Aprendizaje

Incluimos el término enseñanza-aprendizaje porque creemos que es fundamental el procedimiento o la manera de como el docente introduce nuevos conceptos en los estudiantes y cómo es que ellos reaccionan frente a la estrategia aplicada por el docente. Definimos como estrategias de enseñanza todas aquellas propuestas, procedimientos y recursos utilizados por el docente, que proporcionan al estudiante un proceso enseñanza-aprendizaje más eficaz, de tal forma que genere aprendizajes significativos en el estudiante [45].

Sobre este tema, Adorjan y de Kereki (2013) señalan que una buena enseñanza es conseguir que la mayoría de los estudiantes, utilicen el nivel de proceso cognitivo necesario para lograr que los resultados sean óptimos. La estrategia del docente no se trata sólo de las actividades de los docentes, aunque esto sea un componente. En realidad, es un plan para el aprendizaje de otra persona, que abarca las presentaciones que el docente puede hacer, como por ejemplo ejercicios y las actividades diseñadas para los estudiantes y materiales que serán proveídos para que los estudiantes trabajen con ellos [30]. Se han propuesto varios conceptos de enseñanza-aprendizaje en ingeniería.

Estas estrategias puede que motiven a un estudiante y desmotiven a otro [47]. La enseñanza efectiva requiere flexibilidad, creatividad y responsabilidad con el fin de proporcionar un entorno educacional capaz de responder a las necesidades individuales y la capacidad de adaptarse al comportamiento del alumno, y uno de los desafíos que enfrentan los docentes universitarios está relacionado con la respuesta por parte de los estudiantes frente a las estrategias de enseñanza con los estilos de aprendizaje aplicados a los estudiantes con el fin de mejorar los resultados académicos[45]. Los estudiantes tienen diferentes niveles de motivación, actitudes diferentes sobre la enseñanza y el aprendizaje, y diferentes respuestas a ambientes específicos en el aula, prácticas instruccionales, en donde

los docentes entienden las diferencias por ello es la mejor oportunidad que tienen de satisfacer las diversas necesidades de aprendizaje de todos sus estudiantes [46].

Las actividades que requieren que los estudiantes colaboren, compartan soluciones, revisen el trabajo de los demás, o creen materiales han demostrado ser beneficiosos para los estudiantes. En este contexto, la estrategia enseñanza-aprendizaje que utilizaremos se basa tanto en una metodología como las partes que lo componen, que motive eficazmente a los estudiantes en el proceso de aprendizaje como también la participación de los estudiantes en actividades grupales [46].

Algunas de estas estrategias según Orji et al. (2017) son:

- a) **Estrategia de competencia:** Permite a los usuarios competir entre ellos para realizar el comportamiento deseado.
- b) **Estrategia de simulación:** Proporciona los medios para que un usuario observe la relación de causa y efecto de su comportamiento.
- c) **Estrategia de auto-seguimiento:** permite a las personas rastrear sus propios comportamientos, brindando información sobre estados pasados y actuales.
- d) **Estrategia de sugerencia:** sugiere realizar ciertas tareas que orienten a los usuarios durante el uso del sistema.
- e) **Estrategia de elogio:** Felicitar de alguna manera al usuario cuando este logre un comportamiento objetivo. Pueden ser palabras, imágenes, sonidos y símbolos que representen una retroalimentación positiva para el usuario.
- f) **Estrategia de recompensa:** Ofrecer recompensas virtuales por realizar un comportamiento objetivo.

- g) **Estrategia de comparación:** Proporciona un medio para que el usuario vea y compare su desempeño con el rendimiento de otro usuario.
- h) **Estrategia de cooperación:** La cooperación requiere que los usuarios cooperen (trabajen juntos) para lograr un objetivo compartido y obtener recompensas por alcanzar sus metas colectivamente.
- i) **Estrategia de personalización:** Ofrecer contenido y servicios adaptables en función a las características del usuario.

2.6. Metodologías Ágiles (SCRUM)

Hoy en día las metodologías ágiles han demostrado su eficacia en tiempo y recursos, volviéndose en herramientas muy útiles y populares para la industria informática y de desarrollo de software. En el campo de los videojuegos específicamente, muchas compañías enfocadas en el desarrollo de estos suelen emplear dichas metodologías para optimizar y agilizar la preparación de sus entregables; facilitando los famosos “sprints” y ahorrando el tiempo de entrega de estos mismos. Lo que permite esta metodología es garantizar un trabajo ordenado, en el que también se promueve una interactividad colectiva entre los trabajadores logrando mejores resultados. Esto permite, de manera rápida reunir los requerimientos del usuario además de proveer retroalimentación continua durante el desarrollo del software [48]

La metodología SCRUM es de uso común en la actualidad, ya que brinda una aplicación sencilla, escalable e interactiva. SCRUM es mayormente utilizada en el campo de la informática y la programación, exactamente en el desarrollo de software ya que permite una implementación basada en “Sprints” o iteraciones, lo que demanda un menor tiempo de desarrollo. Estos Sprints forman un “Product Backlog”, el cual no es más que un listado de requerimientos o solicitudes identificadas por el “Product owner” y solicitadas

por el cliente. En términos simples, SCRUM es una metodología que propone facilitar el trabajo solicitado por el cliente a través de un equipo SCRUM conformado principalmente por el “Product owner” y el equipo desarrollador. La función del Product Owner es dar a conocer los requerimientos del cliente a través de historias de usuarios y mantener un trabajo dinámico con ayuda de feedback constante y participación entre todo el equipo SCRUM para conseguir el desarrollo final del producto [49].

2.6.1. Roles en SCRUM

Según [49] se define la siguiente terminología :

- a) **Product Owner:** Es el encargado de que el equipo trabaje de forma correcta y esté siempre alineado al negocio y a los requerimientos. El Product Owner ayuda al cliente a definir las historias de usuario, les otorga una prioridad, y las organiza en el Product Backlog bajo su criterio.
- b) **Scrum Master:** También conocido como un facilitador. El Scrum Master es el responsable de que las reglas de la metodología se cumplan. Este se asegura que sean entendidas por el equipo desarrollador y ayuda a superar los obstáculos que imposibilitan la elaboración de un sprint.
- c) **Desarrollador:** También llamado miembro de equipo, según las distintas capacidades estos pueden ser diseñadores gráficos, artistas, programadores, experto en User Experience, etc. En otras palabras, son cada uno de los recursos humanos que realizan la entrega de los Sprints para el desarrollo del producto.
- d) **Stakeholders (Interesados en el proyecto):** Son las personas que hacen que el proyecto sea posible; asimismo, son quienes también reciben el beneficio de este producto. Estos Stakeholders participan directamente durante las revisiones de cada "sprint".

2.6.2. Artefactos o componentes

- a) **Product Backlog:** Registro priorizado de los requerimientos desde la perspectiva del negocio.
- b) **Historias de usuario:** Requerimientos transformados por el Product Owner bajo la perspectiva del Cliente y Stakeholders en Historias.
- c) **Backlog de Sprint:** Conjunto de tareas que se asignan en un determinado sprint a los desarrolladores, estas contienen las especificaciones correspondientes para cada una de ellas.
- d) **Entregable:** Incremento del resultado de cada sprint.





CAPÍTULO 3.
MARCO METODOLÓGICO

1. Alcances y limitaciones

- a) **Viabilidad:** El proyecto en cuanto a las fuentes de datos iniciales, cuenta solo con los datos estadísticos ofrecidos por la oficina de informática de la universidad.
- b) **Lugar:** Las investigación y pruebas se realizaron en los laboratorios de Ingeniería de Sistemas de la Universidad Católica Santa María.
- c) **Licencias de software:** Para el presente estudio se utilizó herramientas de software de acceso gratuito (open source) para el desarrollo de la propuesta con la finalidad de facilitar la continuidad de la investigación en trabajos futuros.
- d) **Financiamiento:** El proyecto fue financiado por el Vicerrectorado de investigación de la Universidad Católica de Santa María, con el contrato N°24162-R-2017- Construcción de estrategias de enseñanza-aprendizaje, basadas en PX (Player eXperience), UX(User eXperience) y TDL(Test-driven Learning), para evitar la deserción de estudiantes de primer año, de la carrera de Ingeniería de Sistemas de la Universidad Católica de Santa María cuyo monto de financiamiento es de S/. 30.000.
- e) **Herramientas:** El presente estudio utilizó como herramienta de medición el dispositivo Emotiv Insight, el cual solo cuenta con 5 canales neuro-sensoriales, lo que limita la obtención de datos y un análisis a profundidad.

2. Aporte

La propuesta del presente trabajo de investigación no pretende aminorar el éxito del enfoque Test Driven Learning demostrado en sus estudios anteriores. Por lo contrario, se busca plantear un método que permita facilitar la adopción del enfoque Test Driven Learning a través de las características lúdicas que ofrecen los Juegos Serios. Permitiendo así continuar con el desarrollo de la investigación sobre TDL en trabajos futuros.

3. Tipo o nivel de investigación

El tipo de investigación según la naturaleza se ha definido como una investigación **cuasi-experimental** dado que el estudio fue aplicado bajo un método de muestreo probabilístico, lo quiere decir que la selección de los estudiantes se realizó de forma predefinida bajo un criterio de aceptación (Estudiantes que cursan el primer y segundo año de la carrera de Ingeniería de Sistemas). La metodología para este tipo de investigación es descriptiva, la cual que se caracteriza por la observación del comportamiento de los datos cualitativos y cuantitativos recabados durante la experimentación con el objetivo de evaluar los cambios que se producen en un periodo de tiempo continuo (Diseño longitudinal). La investigación según su finalidad se define como una investigación aplicada debido a que se plantea una solución a una problemática que presenta el sujeto de estudio, donde los resultados son analizadas y evaluados para comprobar o rechazar la hipótesis propuesta en la investigación, con la finalidad de definir conclusiones que permitan establecer los parámetros y lineamientos para futuras investigaciones.

4. Población y Muestra

- a) **Muestra:** Estudiantes del curso Pensamiento Computacional de primer año y del curso Programación II del segundo año de Ingeniería de Sistemas de la Universidad Católica de Santa María.
- b) **Método de Muestreo:** Probabilístico

5. Métodos, Técnicas e Instrumentos de Recolección de Datos

Para el presente estudio se utilizó los siguientes artefactos para análisis estadístico:

- a) **Datos estadísticos:** Los datos estadísticos iniciales se obtuvieron de la oficina de informática de la Universidad Católica de Santa María. Se utilizan como medio de referencia a la problemática expuesta en la literatura.
- b) **Pre-test y Post-test:** Pruebas de conocimiento elaboradas con la finalidad de evaluar el impacto en el aprendizaje de los estudiantes que participaron del estudio. Estas fueron diseñadas en base al tópico seleccionado (Operaciones de matrices y Transformaciones Geométricas) y se tomaron al comienzo y al final de la experimentación con una duración de 30 min por intento.
- c) **Encuestas:** Encuestas de satisfacción entregadas a los participantes que utilizaron el Juego Serio, permite corroborar los resultados obtenidos de las pruebas de conocimiento. Se aplicó al final de la experimentación con una duración de 30 minutos.
- d) **Base de datos:** Para la recolección de datos en tiempo real con la interacción con la herramienta desarrollada, se utilizó con una base de datos Mysql alojada en un servidor externo para almacenar datos de uso del juego, como lo son: a) el ranking de calificaciones, b) intentos, c) tiempo de juego y d) número de niveles superados.
- e) **Emotiv Insight:** Dispositivo neurosensorial que lee las ondas cerebrales y las traduce en datos numéricos que permiten monitorear cambios en la actividad cerebral (concentración, relajación, compromiso, interés, excitación, afinidad y estrés) provocadas por actividades físicas realizadas por el usuario [50].

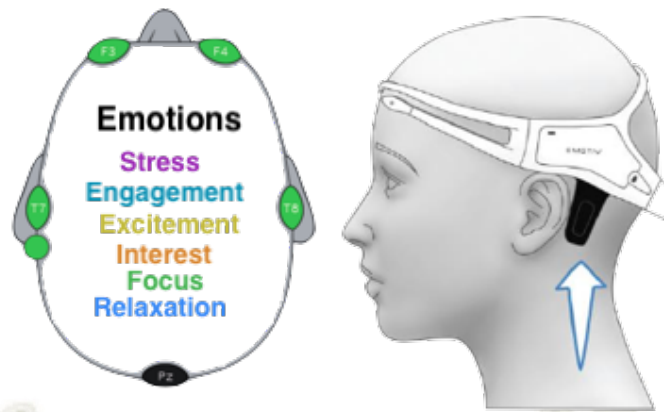


Figura 3.1 Emotiv Insight Headset

Fuente: Manual de usuario Emotiv (2017)

Como se puede ver en Figura 3.1 este dispositivo electrónico inalámbrico, a través de sus sensores permite leer en tiempo real las ondas cerebrales, traducirlas en datos significativos, rastrear y monitorear la actividad cerebral. Estos datos son visualizados en 6 métricas cognitivas clave a través de su aplicativo oficial “MyEmotiv” a través de la tienda de aplicaciones del mismo proveedor.

6. Plan de análisis estadístico de los datos

Se han elegido los siguientes procedimientos e instrumentos estadísticos con el fin de evaluar del Juego Serio en cuanto a conocimientos adquiridos, para así realizar un contraste con los resultados recabados de la literatura.

- a) Procedimientos estadísticos:** El proceso de análisis consta de verificación de la muestra obtenida, para determinar si presenta normalidad en los datos para ello se aplica una prueba de *Kolmogorov-Smirnov*. Si los datos presentan normalidad se procede a aplicar la prueba T-student, en caso contrario se aplica la prueba de *Wilcoxon* de los rangos con signo para muestras no paramétricas. Esto para determinar si existen diferencias entre las muestras. Posteriormente para reforzar las conclusiones de los resultados, se procede a validar las correlaciones con otras variables del estudio

como: a) El ranking, b) El tiempo de juego y c) El número de niveles jugados. Para determinar ello se aplica el método de *Pearson* en el caso de datos paramétricos y *Spearman* para datos no paramétricos.

b) Herramientas Analíticas: Se utilizó la herramienta SPSS Statistics para el análisis estadístico de licencia de prueba.

7. Metodología de desarrollo escogida

Para el presente trabajo de investigación se decidió usar SCRUM como metodología de desarrollo, por sus ventajas en el desarrollo de videojuegos [48]. Se proyectó 3 Sprints de 2 semanas cada una para culminar con la primera versión de la herramienta. Para poder desarrollar la propuesta se tuvo que seleccionar cuidadosamente una metodología que sea apropiada para un proyecto con un alcance corto como este y un equipo pequeño. Se enfatizan en estos dos factores debido a que, en la industria de los videojuegos, es común tener proyectos donde participan más de cien personas con presupuestos que pueden ser de varias decenas de millones de dólares. Por ello, es importante notar el contexto dentro del cual los autores plantean sus metodologías, antes de tomar una decisión.

8. Aspectos Disciplinarios

En esta sección se presentan una serie de conceptos, explicaciones y ejemplos utilizados para la construcción del contenido académico del Juego Serio (Engineers Escaping), el cual se ha estructurado en base a los materiales de estudio: a) “Cálculo de una variable”[51] del curso de Álgebra y Geometría Analítica y b) “The C++ Programming Language 4th Edition” [52] del curso Programación I y II , los cuales generalmente son dictados en el primer y segundo semestre de la carrera de Ingeniería de sistemas.

8.1. Matrices y operaciones básicas

Según Groosman (2012) una matriz es un arreglo ordenado de valores. Una matriz es un conjunto rectangular de números con m filas y n columnas, se puede denotar como $A_{m,n}$ donde $m \times n$ son sus dimensiones.

$$A_{m,n} = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix}$$

a) Matriz cuadrática: Podemos decir que una matriz es cuadrática cuando el número de filas (m) es igual número de columnas (n) y se puede denotar como $A_{n \times n}$.

Ejemplo:

$$P_{2 \times 2} = \begin{pmatrix} 1 & -3 \\ 2 & 4 \end{pmatrix} \mid M_{3 \times 3} = \begin{pmatrix} 1 & 3 & 4 \\ 3 & 2 & 2 \\ -2 & 1 & 3 \end{pmatrix}$$

P es una matriz cuadrática de orden = 2 y M es una matriz cuadrática de orden = 3:

b) Matriz transpuesta: Una matriz transpuesta se obtiene intercambiando sus filas (m) por sus columnas (n). Generalmente se denota como A^t .

Ejemplo:

$$P = \begin{pmatrix} 3 & 4 \\ -2 & 6 \\ 1 & 8 \end{pmatrix} \mid P^t = \begin{pmatrix} 3 & -2 & 1 \\ 4 & 6 & 8 \end{pmatrix}$$

P es una matriz de orden (3x2), su transpuesta P^t es una matriz de orden (2x3).

c) Igualdad de matrices: Se puede decir que una matriz es igual a otra cuando esta tiene la misma dimensión y sus valores son los mismos en la misma posición.

Ejemplo:

$$P = \begin{pmatrix} 1 & 5 & 1 \\ 2 & 3 & -2 \\ -3 & 4 & 5 \end{pmatrix} \mid M = \begin{pmatrix} 1 & 5 & 1 \\ 2 & 3 & -2 \\ -3 & 4 & 5 \end{pmatrix}$$

P y M son matrices de orden (3x3) y tienen los mismos elementos en la misma ubicación. Por lo tanto, podemos decir que P es igual a M.

d) Matrices en programación: En programación una matriz generalmente se utiliza como una estructura de almacenamiento contiguo de valores del mismo valor, esta se define como un arreglo bidimensional la cual se conforma de un vector que a su vez contiene una serie de vectores [52]. Como se puede observar en la Figura 3.2 que se muestra a continuación:

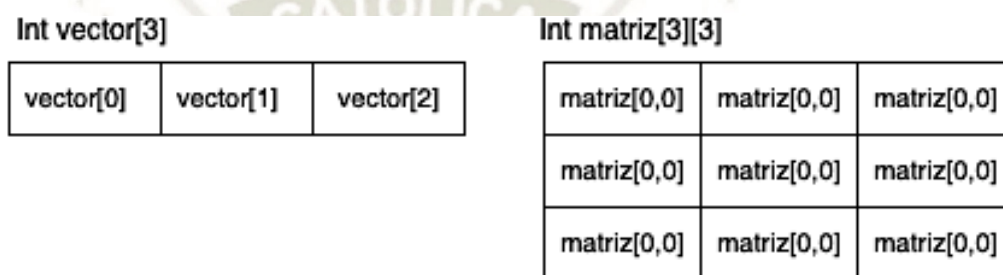


Figura 3.2 Representación de matrices en Programación

Fuente: Elaboración propia (2018)

8.1.1. Algebra de matrices: Suma y Resta

Dos matrices: $A \pm B = [a_{ij} \pm a_{ij}]$ se pueden sumar o restar si tienen la misma dimensión. La operación $A \pm B$ es la matriz m x n obtenida al sumar o restar las entradas correspondientes de A ($a_{ij} \dots$) y B ($b_{ij} \dots$). Se puede representar de manera extendida se la siguiente forma:

$$\begin{pmatrix} a_{00} & \cdots & a_{0n} \\ \vdots & \ddots & \vdots \\ a_{m0} & \cdots & a_{mn} \end{pmatrix} \pm \begin{pmatrix} b_{00} & \cdots & b_{0n} \\ \vdots & \ddots & \vdots \\ b_{m0} & \cdots & b_{mn} \end{pmatrix} = \begin{pmatrix} [a_{00} + b_{00}] & \cdots & [a_{0n} + b_{0n}] \\ \vdots & \ddots & \vdots \\ [b_{m0} + a_{m0}] & \cdots & [a_{mn} + b_{mn}] \end{pmatrix}$$

A continuación, se describe el algoritmo clásico para sumar dos matrices de manera iterativa (A y B).

```

Funcion Suma_de_matrices (A, B, S Por Referencia)
    // m = columnas de B
    // n = filas de A
    Para i ← 1 Hasta m Hacer
        Para j ← 1 Hasta n Hacer
            S[i,j] ← A[i,j] + B[i,j]
        FinPara
    FinPara
FinFuncion
    
```

Figura 3.3 Algoritmo de suma de matrices

Fuente: Elaboración propia (2018)

Como se puede observar en la Figura 3.3, el procedimiento se basa en el uso de dos ciclos anidados. En cada iteración se almacena el resultado de la suma de los elementos de las matrices A y B en sus índices correspondientes sobre la matriz S.

8.1.2. Multiplicación de matrices:

Dos matrices pueden multiplicarse si el número de columnas de A coincide con el número de filas de B. Esta operación se denota generalmente como $A \times B$ o simplemente AB . Donde a) $m \leftarrow$ El número de filas de la matriz A, b) $n \leftarrow$ El número de columnas de la matriz A y número de filas de la matriz B y c) $k \leftarrow$ El número de columnas de B.

$$\begin{array}{ccc}
 \text{Matriz A} & & \text{Matriz B} & & \text{Matriz C} \\
 \begin{bmatrix} a_{00} & \dots & a_{0n} \\ a_{10} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m0} & \dots & a_{mn} \end{bmatrix} & \times & \begin{bmatrix} b_{00} & b_{01} & \dots & b_{0k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n0} & b_{n1} & \dots & b_{nk} \end{bmatrix} & = & \begin{bmatrix} c_{00} & c_{01} & \dots & c_{0k} \\ c_{10} & c_{11} & \dots & c_{1k} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m0} & c_{m1} & \dots & c_{mk} \end{bmatrix} \\
 m \times n & & n \times k & & m \times k
 \end{array}$$

Figura 3.4 Multiplicación de dos matrices

Fuente: Elaboración propia (2018)

La multiplicación de matrices es una de las operaciones del álgebra matricial, que a diferencia de la suma no cumple con la propiedad de conmutatividad ($AB \neq BA$). Como se muestra en la figura anterior el producto de las matrices A y B da como resultado la matriz C. Esta operación que se denota como $C = AB := (C_{ij})_{m \times p}$ y cada elemento de C está definido como:

$$C_{ij} = \sum_{k=1}^n a_{ik} * b_{kj}$$

Y como puede observarse en esta notación, la matriz C solo está definida cuando el número de columnas de A es igual al número de filas en B (k).

A continuación, se describe el algoritmo clásico para sumar dos matrices en pseudocódigo.

```

Funcion Mult_de_matrices (A, B, R Por Referencia)
  // m = columnas de B
  // p = filas de B
  // n = filas de A
  Para i ← 1 Hasta m Hacer
    s ← 0
    Para j ← 1 Hasta p Hacer
      Para k ← 1 Hasta n Hacer
        s ← A[i,k] * B[k,j] + s
      FinPara
      R[i,k] ← s
    FinPara
  FinPara
FinFuncion

```

Figura 3.5 Algoritmo para multiplicar dos matrices

Fuente: Elaboración propia (2018)

8.2. Transformaciones geométricas

Las transformaciones son procedimientos para calcular nuevas posiciones de estos píxeles, cambiando el tamaño y orientación de la imagen. Entonces si P es un

punto de la imagen, al cual se le aplica una transformación T, P' es el mismo punto, pero transformado [54].

$$T(P) = P'$$

Entre las transformaciones básicas, tenemos las siguientes: a) Traslación: Son movimientos directos en la posición (x, y) sin cambiar la orientación, la forma y el tamaño de la imagen, b) Rotación: Transformación que permite modificar la orientación de la imagen, manteniendo su centro (posición) original, y c) escalamiento, esta transformación incrementa el tamaño de una imagen, escalando las dimensiones y la posición de la imagen [54].

a) Traslación geométrica:

Dado que cada valor se ha transformado como lo describe la siguiente ecuación para una traslación lineal.

$$P' = \begin{cases} x' = x + t_x \\ y' = y + t_y \end{cases}$$

Su representación en forma matricial es la siguiente:

$$P = (x, y) \quad P' = (x', y') \quad T = (t_x, t_y)$$

$$P' = P + T$$

b) Rotación geométrica:

Una rotación geométrica se basa en el ángulo de inclinación la representación matricial es la siguiente:

$$P' = P * R(\theta)$$

Para realizar una rotación geométrica con respecto al origen se aplicaría la siguiente fórmula:

$$(x', y', 1) = (x, y, 1) \begin{pmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

c) Escalamiento geométrico:

Un escalamiento geométrico, permite realizar un incremento de las dimensiones de un objeto, se basa en la multiplicación de dos factores s_1 y s_2 sobre los ejes x y y .

$$P' = P * S(s)$$

Para realizar un escalamiento geométrico se puede representar de la siguiente forma:

$$(x', y', 1) = (x, y, 1) \begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

9. Aspectos pedagógicos

En esta sección se describen los mecanismos habituales de la docencia, aspectos pedagógicos en la enseñanza de cursos de programación y matemáticas en base la experiencia de los docentes y alumnos de la carrera de Ingeniería de sistemas. Así mismo, se presenta una propuesta pedagógica alternativa que permita la ejecución del Juego Serio en la etapa de experimentación.

9.1. Pedagogía tradicional

La pedagogía en cursos de programación y matemáticas en sus primeras etapas suele ser un desafío en la docencia. Por lo general es común utilizar herramientas conservadoras tales como diapositivas, videos, ejercicios y el propio material del curso. Usualmente, el docente inicia con la teoría y relaciona la práctica a través de ejemplos. La mayoría de los conceptos que se presentan son nuevos para los estudiantes, por lo que la curva de aprendizaje puede llegar a ser agresiva en esta etapa.

En el curso de fundamentos matemáticos (Álgebra y geometría) los estudiantes por lo general no presentan muchas dificultades, dado que son conocimientos ya adquiridos en la etapa escolar (Nivel secundario). Sin embargo, en los cursos de programación se asume que los estudiantes ya dominan estas bases matemáticas, por lo que el docente enfoca el desarrollo del curso solo a la programación y a la lógica algorítmica. Es aquí donde surge la problemática de muchos estudiantes que se encuentran en un proceso de nivelación en sus conocimientos matemáticos. Debido a estas brechas en sus conocimientos, presentan dificultades en su aprendizaje al no lograr relacionar la teoría con la práctica en programación.

9.2. Pedagogía propuesta

Para la propuesta del presente trabajo, se optó por considerar un recurso alternativo que promueva una pedagogía basada en características lúdicas. Para ello, se utilizó el material propuesto en CS Education Research Group (2014) Representación de imágenes bajo el nombre “Colors by Numbers”, el cual consiste en almacenar dibujos utilizando sólo números en una matriz la cual puede ser representada de forma gráfica en una matriz de píxeles, como lo representaría una computadora. Esto permite asociar de manera aplicada el tópico seleccionado (Matrices y Transformaciones geométricas) al Juego Serio desarrollado. Una manera de enseñar el procesamiento de imágenes en un computador es través de la representación de imágenes en forma de matrices, una imagen contiene una gran cantidad de datos internamente, es por ello que necesita ser comprimida en función de píxeles como bloques de construcción para ser almacenada y transferida de manera eficiente, la manera más básica de hacerlo es a través de una compresión de dos dimensiones, conocida como una compresión de Grupo 3 según la CCITT (International Consultative Committee for Telegraph and Telephone) para una imagen en blanco y negro, (como lo haría una máquina de fax), esta codificación puede representarse en una matriz

numérica[54]. Sobre esto la CS Education Research Group (2014) presenta una actividad didáctica y entretenida, la cual podemos considerar un Game Based Learning por ser únicamente presencial, en su portal podemos encontrar un manual de como implementarlo correctamente. La actividad es simple, inicialmente por un lado tenemos una grilla de píxeles del otro tenemos una representación numérica de dos dimensiones (una matriz), donde cada columna alternada por color, indica el número de píxeles que se deben colorearse (blanco, negro, blanco, etc.).

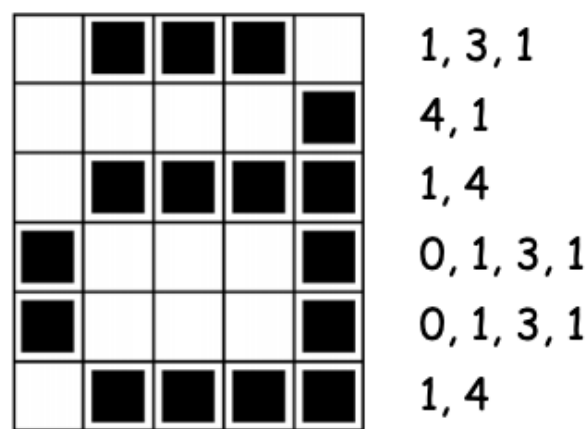


Figura 3.6 Representación de imágenes por compresión de dos dimensiones

Fuente: Csunplugged.org (2014)

Como se muestra en la Figura 3.6 , una imagen puede ser representada por números en una matriz bidimensional. Donde, la primera fila consta de un píxel blanco, luego tres negros, luego uno blanco, representada como: 1, 3, 1.



Figura 3.7 Actividad presencial: Representación de imágenes

Fuente: Csunplugged.org (2014)

La actividad original se basa en colorear algunas docenas de píxeles negros con un lápiz, en tiempo real. Una forma de hacerlo es utilizar una lata de pintura en aerosol negra y una plantilla cuadrada para crear una imagen gigante píxel por píxel. Otra forma es procesar una fotografía de ejemplo y hacer que una clase completa de estudiantes decodifique la imagen oculta.

Existen algunas variaciones de esta actividad, pero en esencia es la misma, es posible utilizar otros materiales adicionales para mejorar la experiencia de juego, como, por ejemplo:

- Utilizar la cuadrícula con cuadros de papel adhesivo para poder corregir errores.
- En el caso de ser una actividad en papel, utilizar papel de calcar debajo de la rejilla para así tener una imagen más clara.



CAPÍTULO 4.
DESARROLLO DE LA PROPUESTA

1. Análisis de software

Como menciona Hernández et al. (2019) , la etapa de análisis está circunscrita a la fase de pre-producción en el desarrollo de videojuegos. En este capítulo se presenta la metodología que se usa para todo el resto del desarrollo de los componentes que corresponden al alcance de este proyecto. Además, se presentan los requisitos del sistema junto con el análisis de la solución propuesta.

1.1. Análisis de requerimientos

Una de las grandes diferencias entre el desarrollo de software tradicional y el desarrollo de Juegos Serios es el proceso de toma de requisitos o educación de requerimientos. En este tipo de proyectos, el analista debe ir un paso adelante y evitar el proceso tradicional de elicitación de requerimientos para dar lugar a un proceso creativo de definición de los mismos acorde con los objetivos de aprendizaje [55], el cual opcionalmente podría ser retroalimentado con alguna muestra de usuarios [44].

1.2. Objetivos de aprendizaje

Tabla 4.1

Objetivos de aprendizaje

ID	Requerimiento	Conocimiento requerido
AO01	Representar una matriz numérica a su representación gráfica (píxeles)	Ninguno
AO02	Resolver operaciones matriciales (Suma y Multiplicación)	Matrices
AO03	Utilizar coordenadas geométricas para aplicar transformaciones geométricas.	Operaciones matriciales
AO04	Diseñar casos de prueba para evaluar los algoritmos de Suma y Multiplicación de matrices	Pruebas unitarias y Testing
AO05	Diseñar los algoritmos Suma y Multiplicación de Matrices	Programación Bloques

Elaboración propia

1.3. Requerimientos Funcionales

Los requerimientos funcionales describen el comportamiento del software en cuestión en base a los objetivos de aprendizaje. En el ámbito de los Juegos Serios pueden clasificarse según Spinelli & Massa (2018) como:

- a) **Requerimientos Pedagógicos:** Se refieren aquellos aspectos que promueven el aprendizaje y el planteamientos de las estrategias de enseñanza, se relacionan directamente con los objetivos de aprendizaje.
- b) **Requerimientos de Dominio:** Son aquellos que definen la mecánica del Juego Serio (Vistas, Artefactos, Personajes, etc), le dan forma a los aspectos didácticos a través de la trama del juego.
- c) **Requerimientos del Juego:** Estas se definen de acuerdo a la narrativa y las características y dinámicas generales del género del Juego.

A continuación, se presentan los requerimientos funcionales en forma de historias de usuario, estas se han obtenido a través la literatura, los objetivos del proyecto y la narrativa propuesta.

Tabla 4.2
Historia de usuario 01 – Menú principal

HU01	Menú principal	Tipo: Dominio
Como jugador quiero poder ver la pantalla de inicio al momento de iniciar el juego para poder seleccionar la opción correspondiente.		
Prioridad: 2	Sprint: 1	Dependencias: Ninguna
Condiciones de validación		
<ul style="list-style-type: none"> • Antes de iniciar el juego se debe visualizar la pantalla de inicio con el título del juego: “Engineers escaping”, y las siguientes opciones para el jugador: a) Jugar, b) Créditos y c) Salir 		

Elaboración propia

Tabla 4.3

Historia de usuario 02 – Historia de juego

HU02	Historia de juego	Tipo: Dominio
Como jugador quiero visualizar de manera audiovisual la historia del juego para entender el objetivo del juego.		
Prioridad: 4	Sprint: 2	Dependencias: Ninguna
Condiciones de validación		

- Después de haber iniciado el juego (Opción Jugar) se debe narrar la historia de manera gráfica, a través de audios y globos de texto.

Elaboración propia

Tabla 4.4

Historia de usuario 03 – Autenticación de usuario

HU03	Autenticación del usuario	Tipo: Dominio
Como jugador quiero poder iniciar sesión a través de mi código de estudiante para obtener mis datos y empezar el juego		
Prioridad: 5	Sprint: 3	Dependencias: HU02
Condiciones de validación		

- Finalizada la narración del juego (HU02), se debe solicitar al jugador que ingrese su código universitario (10 dígitos) el cual debe recuperar el nombre del estudiante y su progreso.
- Se debe validar la comunicación con la base de datos en el servidor externo.

Elaboración propia

Tabla 4.5

Historia de usuario 04 – Niveles de juego

HU04	Niveles de juego	Tipo: Dominio
Como jugador quiero poder visualizar todos los niveles (Capítulos) para poder seleccionar y jugar alguno de ellos.		
Prioridad: 3	Sprint: 1	Dependencias: Ninguna

Condiciones de validación

- Cada vez que el jugador avance de nivel se debe mostrar en forma de recorrido todos los niveles (Capítulos) del juego.
- Cuando el jugador seleccione algún capítulo en particular debe redirigirlo al escenario correspondiente
- Cada ítem de nivel debe tener asociado el puntaje obtenido en caso de haberlo jugado previamente

Elaboración propia

Tabla 4.6

Historia de usuario 04 – Gameplay

HU05

Gameplay

Tipo: Juego

Como jugador al iniciar por primera vez quiero poder visualizar información de los controles (Gameplay) para saber cómo interactuar con el juego.

Prioridad: 4

Sprint: 2

Dependencias: Ninguna

Condiciones de validación

- Al comienzo del juego (Nivel de introducción) se debe mostrar una pantalla con la descripción de todos los controles y sus acciones correspondientes.

Elaboración propia

Tabla 4.7

Historia de usuario 06 –Abrir una compuerta

HU06

Abrir una compuerta (Puzle)

Tipo: Juego

Como jugador quiero poder desactivar el cerrojo de las compuertas para poder seguir avanzando en el juego

Prioridad: 1

Sprint: 1

Dependencias: Ninguna

Condiciones de validación

- El jugador al pulsar la tecla “E” estando en frente de una compuerta este será direccionado al puzle que desarma el cerrojo, estos puzles de lógica pueden ser: a) Representación de imágenes, b) Trivia de matrices, c) Transformación geométrica, d) Rotación geométrica y e) Escalamiento geométrico.

- Se debe mostrar en la esquina superior derecha un menú desplegable con las siguientes: a) Activar: Verifica la respuesta del jugador con la solución, b) Resolver: Aplica una transformación geométrica (de ser necesario) y c) Salir: Oculta el menú

Elaboración propia

Tabla 4.8

Historia de usuario 07 – Resolver un puzle de representación de imágenes.

HU07 Resolver un puzle de representación de imágenes Tipo: Pedagógico

Como jugador quiero poder leer una matriz numérica y representarla de manera gráfica en una matriz de píxeles para desactivar el cerrojo de la compuerta.

Prioridad: 1 **Sprint:** 1 **Dependencias:** Ninguna

Condiciones de validación

- Se debe mostrar en forma de pergamino desplegable del lateral izquierdo una matriz numérica, así como links e información extra en la parte inferior que faciliten el planteamiento del ejercicio.
- Al hacer click sobre cada recuadro de la matriz de píxeles, debe cambiar de color de blanco a negro y viceversa.
- La matriz gráfica debe sincronizarse con su matriz numérica, con la finalidad de verificar con la matriz resultado.

Elaboración propia

Tabla 4.9

Historia de usuario 08 – Resolver un puzle de transformación geométrica

HU08 Resolver un puzle de transformación geométrica Tipo: Pedagógico

Como jugador quiero aplicar una transformación geométrica para resolver el puzle y avanzar en el juego.

Prioridad: 1 **Sprint:** 1 **Dependencias:** HU07

Condiciones de validación

- Se debe mostrar en el pergamino las instrucciones de la transformación, así como cualquier información adicional.

- La matriz de píxeles no debe permitir modificaciones, esto quiere decir que no se puede re colorear directamente en la imagen.
- Al pulsar “Resolver” en el menú del puzzle debe mostrar la opción para aplicar una transformación (Traslación, Rotación y Escalamiento).

Elaboración propia

Tabla 4.10

Historia de usuario 09 – Resolver un puzzle de traslación geométrica

HU09 Resolver un puzzle de traslación geométrica Tipo: Pedagógico

Como jugador quiero poder aplicar una TRASLACIÓN geométrica para mover una imagen y así resolver el puzzle y avanzar en el juego.

Prioridad: 2 Sprint: 1 Dependencias: HU08

Condiciones de validación

- Al seleccionar TRANSLACIÓN de las opciones disponibles, se debe mostrar el procedimiento: Una suma de matrices entre la matriz principal + matriz de transformación = matriz resultante.
- Al realizar la suma correctamente, debe visualizarse el movimiento en el eje x en la matriz gráfica (Imagen)

Elaboración propia

Tabla 4.11

Historia de usuario 10 – Resolver un puzzle de rotación geométrica

HU10 Resolver un puzzle de rotación geométrica Tipo: Pedagógico

Como jugador quiero poder aplicar una ROTACIÓN geométrica para girar una imagen en un ángulo determinado y así resolver el puzzle y avanzar en el juego.

Prioridad: 4 Sprint: 2 Dependencias: HU08

Condiciones de validación

- Al seleccionar ROTACIÓN debe solicitarse el ángulo de inclinación, la rotación se realizará internamente.

- Al ingresar el valor correcto, se debe visualizar la rotación en la matriz gráfica (Imagen)

Elaboración propia

Tabla 4.12

Historia de usuario 11 – Resolver un puzzle de escalamiento geométrica

HU11 Resolver un puzzle de escalamiento geométrica Tipo: Pedagógico

Como jugador quiero poder aplicar un ESCALAMIENTO geométrico para aumentar el tamaño de una imagen y así resolver el puzzle y avanzar en el juego.

Prioridad: 3 Sprint: 2 Dependencias: HU08

Condiciones de validación

- Al seleccionar ESCALAMIENTO de las opciones disponibles (Resolver) debe mostrarse el procedimiento: Matriz de la imagen + Matriz de escalamiento = Matriz resultado.
- Al ingresar el valor correcto, se debe visualizar el incremento de tamaño en la Imagen

Elaboración propia

Tabla 4.13

Historia de usuario 12 – Resolver un puzzle de trivia

HU12 Resolver un puzzle de trivia Tipo: Pedagógico

Como jugador quiero poder resolver una trivia sobre matrices, seleccionar la opción correcta para avanzar en el juego

Prioridad: 5 Sprint: 3 Dependencias: HU06

Condiciones de validación

- Se debe mostrar el enunciado de la trivia y una lista de opciones
- Al pulsar en “Activar”, se debe validar si la respuesta seleccionada es correcta.

Elaboración propia

Tabla 4.14

Historia de usuario 13 – Ver cronómetro en los puzzles

HU13 Ver cronómetro en los puzles

Tipo: Dominio

Como jugador deseo poder ver un cronómetro en los puzles para poder medir el tiempo disponible que tengo para resolverlos.

Prioridad: 2

Sprint: 1

Dependencias: HU06

Condiciones de validación

- Se debe visualizar un cronómetro para cada puzle, este debe ser programable internamente para cada uno de los tipos de puzles del juego.
- Cuando el cronómetro llegue a 00:00 se debe detener el juego y mostrar una ventana de “Tiempo completado”

Elaboración propia

Tabla 4.15

Historia de usuario 14 – Obtener puntaje

HU14

Obtener puntaje

Tipo: Dominio

Como jugador quiero obtener un puntaje cada vez que resuelva un puzle para así poder validar mi progreso y mi calificación final.

Prioridad: 3

Sprint: 2

Dependencias: HU06, HU11

Condiciones de validación

- Cada que se resuelva un puzle correctamente, el puntaje se debe calcular del tiempo restante en relación con el máximo puntaje posible.
- El puntaje debe ser acumulativo y debe mostrarse siempre en el juego (parte superior derecha de la pantalla).

Elaboración propia

Tabla 4.16

Historia de usuario 15 – Recoger pistas (Hints)

HU15 **Recoger pistas (Hints)** **Tipo: Juego**

Como jugador quiero poder recoger pistas que me brinden información relevante para saber sobre cómo resolver los puzles de las puertas.

Prioridad: 4 **Sprint: 3** **Dependencias: Ninguna**

Condiciones de validación

- Se debe obtener por contacto con el jugador.
- Al recoger alguna pista debe mostrarse una notificación al jugador.

Elaboración propia

Tabla 4.17

Historia de usuario 16 – Leer las pistas (Hints)

HU16 **Leer las pistas o Hints** **Tipo: Juego**

Como jugador quiero ver mis pistas para tenerlo disponible en cualquier parte del juego.

Prioridad: 3 **Sprint: 2** **Dependencias: HU12**

Condiciones de validación

- Las pistas se deben visualizar en una grilla desde el menú de pausa del juego.
- Se deben utilizar las direccionales (Arriba y abajo) para desplazarse en el pergamino de cada pista.

Elaboración propia

Tabla 4.18

Historia de usuario 17 – Grabar el progreso de avance

HU17 **Grabar el progreso de avance** **Tipo: Dominio**

Como jugador quiero grabar mi progreso y mi avance en un punto de control (Check point) del recorrido para no empezar desde el inicio del nivel nuevamente.

Prioridad: 5 **Sprint: 3** **Dependencias: Ninguna**

Condiciones de validación

- Los puntos de control se deben activar cuando el jugador cruza en frente de estos.
- Se debe notificar al usuario cuando se alcanzó algún punto de control.

Elaboración propia

Tabla 4.19

Historia de usuario 18 – Ver puntaje final

HU18 **Ver puntaje final** **Tipo: Dominio**

Como jugador quiero ver el resumen al culminar cada nivel para saber cuál es mi puntaje final y el tiempo de juego.

Prioridad: 3 **Sprint: 2** **Dependencias: HU12**

Condiciones de validación

- Los puntos de control se deben activar cuando el jugador cruza en frente de estos.
- Se debe notificar al usuario cuando se alcanzó algún punto de control.

Elaboración propia

Tabla 4.20

Historia de usuario 19 – Recolectar recompensas (Gemas)

HU19 **Recolectar recompensas (Gemas)** **Tipo: Juego**

Como jugador quiero recolectar gemas para poder acumularlas y utilizarlas posteriormente.

Prioridad: 5 **Dificultad: 3** **Dependencias: Ninguna**

Condiciones de validación

- Las gemas deben ser flotantes y fácilmente visibles
- El contador de gemas debe visualizarse en la parte superior derecha

Elaboración propia

Tabla 4.21

Historia de usuario 20 – Denegar el acceso al juego

HU20 **Denegar el acceso al juego** **Tipo: Dominio**

Como gestor quiero desactivar el juego remotamente para evitar que pueda ejecutarse fuera del campo de experimentación.

Prioridad: 4 **Sprint: 3** **Dependencias: Ninguna**

Condiciones de validación

- Se debe mostrar una pantalla de acceso restringido cuando la configuración general del juego indique no estar disponible.

Elaboración propia

1.4. Requerimientos NO Funcionales

En esta sección se presentan los requerimientos no funcionales, representan características generales y lineamientos para Juego Serio.

Tabla 4.22

Requerimiento No Funcional 01 – Seguridad y control de acceso

Requerimiento No Funcional N° 1

ID	RNF01
Título	Seguridad y control de acceso del Juego Serio
Descripción	Dado que el juego se encontraba en una fase para la obtención de derechos de autor, este debía ser estrictamente permitido su uso dentro del campus universitario. Es por ello que se deben tomar medidas tales como autenticación y control de acceso de forma remota.

Prioridad: 1

Condiciones de validación

- Se debe validar que el juego no permite la autenticación fuera del campo de aplicación.

Elaboración propia

Tabla 4.23

Requerimiento No Funcional 02 Usabilidad del Juego Serio

Requerimiento No Funcional N° 2

ID	RNF02
Título	Usabilidad
Descripción	Un Juego Serio para que sea efectivo debe presentar una interfaz intuitiva, de fácil aprendizaje.

Prioridad: 3

Condiciones de validación

- Se deben realizar pruebas alfa previas al lanzamiento para detectar falencias en la usabilidad del Juego Serio.

Elaboración propia

Tabla 4.24

Requerimiento No Funcional N°3: Multiplataforma (Portabilidad)

Requerimiento No Funcional N° 3

ID	RNF03
Título	Multiplataforma
Descripción	El Juego Serio debe ser capaz de ejecutarse en sistemas operativos Mac OSX (para pruebas de desarrollo), Windows 8-10, Web (Chrome y Firefox) dado la limitante en el tiempo de preparación e instalación en los equipos de la universidad

Prioridad: 3

Condiciones de validación

- Se debe validar que el tiempo de instalación y ejecución sea corto
- Para aquellos equipos que no presenten las características de hardware necesarias para su ejecución, se debe validar el funcionamiento desde la web (Google Chrome y Firefox)

Elaboración propia

2. Diseño de Software

El diseño de software se realizó en base al modelo de vistas de arquitectura 4+1 el cual permite describir la arquitectura de Software en 5 concurrencias o perspectivas [56], para realizar el modelamiento de cada una de estas se utilizó la herramienta StarUML.

2.1. Vista de Escenarios (+1)

Esta vista describe la relación y las secuencias entre los objetos y procedimientos con la finalidad de validar el diseño de arquitectura de software.

A continuación, se presentan los casos de uso en relación con el rol dentro del Juego Serio. Esta vista contempla la siguiente estructura: a) Identificador, b) Título, c) Descripción (Flujo principal), d) Referencias (Historias de usuario) y e) Condiciones de validación.

Tabla 4.25

Caso de uso N°1: El jugador inicia una nueva partida

Caso de uso N° 1

ID	CU01
Título	El jugador inicia una nueva partida
Descripción	<ul style="list-style-type: none"> - El jugador inicia el juego por primera vez - Selecciona la opción “Jugar” - El jugador inicia una nueva partida. - El jugador visualiza la historia que introduce el juego. - Al finalizar el usuario se autentifica con su código universitario. - Inicia el primer nivel
Referencias:	3
Condiciones de validación	
<ul style="list-style-type: none"> • Se debe validar que la historia de usuario inicie siempre que el usuario interactúe con el juego por primera vez. • En el proceso autenticación se debe obtener automáticamente los datos del estudiante, para ello deben estar registrados previamente. 	

Elaboración propia

Tabla 4.26

Caso de uso N°2: El jugador carga una partida anterior

Caso de uso N° 2

ID	CU02
Título	El jugador carga una partida anterior
Descripción	<ul style="list-style-type: none"> - El jugador inicia el juego por segunda vez. - Selecciona la opción “Jugar”, - El juego obtiene al usuario autenticado. - El jugador visualiza todos los niveles que ha jugado. - El jugador selecciona el nivel que desea jugar. - El jugador inicia el juego.
Referencias:	3
Condiciones de validación	

- Se debe obtener los datos del usuario y todo su progreso para cargar la partida, en caso el juego ha sido detenido e iniciado nuevamente, el jugador debe realizar el proceso de autenticación nuevamente

Elaboración propia

Tabla 4.27

Caso de uso N°3: El jugador visualiza el ranking general

Caso de uso N° 3

ID	CU03
Título	El jugador visualiza el ranking general
Descripción	<ul style="list-style-type: none"> - El jugador inicia el juego y selecciona la opción “Jugar” - El jugador visualiza todos los niveles que ha jugado. - El jugador selecciona la opción de ranking - Se muestra un listado de todos los jugadores y sus puntajes
Referencias:	3
Condiciones de validación	

- Se debe obtener el listado de los puntajes actualizados de todos jugadores desde un recurso JSON/API de la base de datos

Elaboración propia

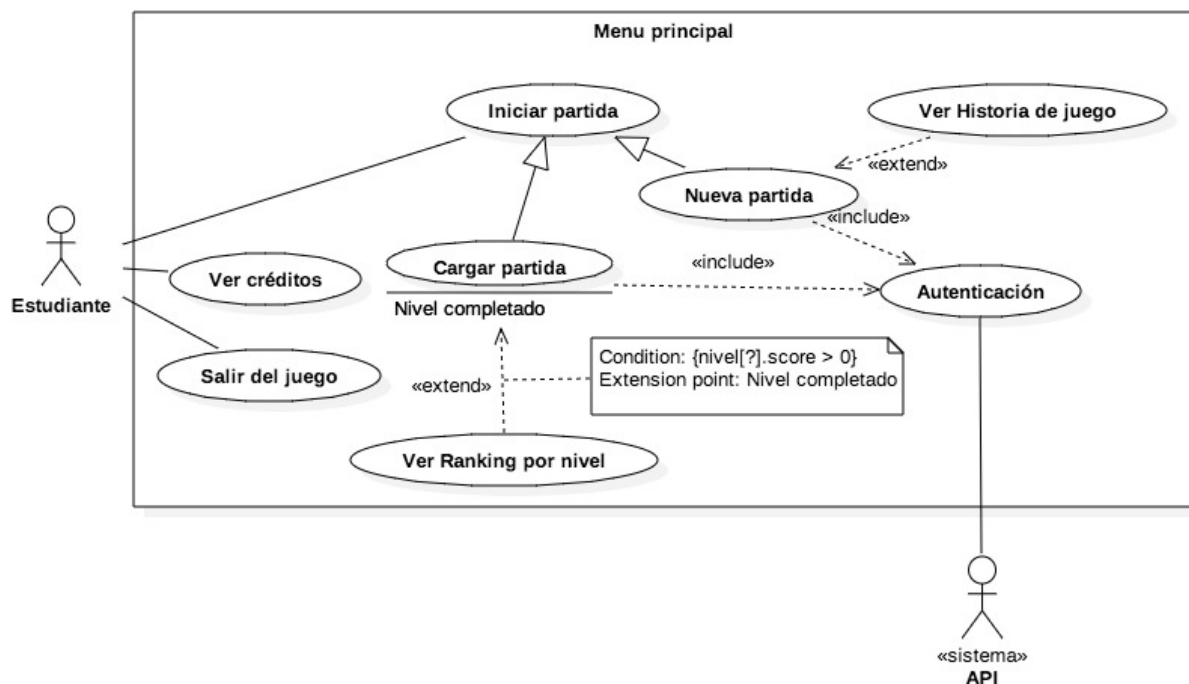


Figura 4.1 Vista de casos de uso Jugador Inicio del juego

Fuente: Elaboración propia (2018)

Tabla 4.28

Caso de uso N°4: El jugador carga una partida anterior

Caso de uso N° 4

ID	CU04
Título	El jugador recolecta pergaminos informativos
Descripción	<ul style="list-style-type: none"> - El jugador selecciona e inicia un nivel - Durante el recorrido al jugador se le presenta un objeto en forma de pergamino - El jugador colisiona con el objeto - El jugador es notificado del nuevo recurso obtenido

Referencias: 3

Condiciones de validación

- Finalizado el proceso, el jugador debe poder visualizar el pergamino informativo desde el menú de pausa.

Elaboración propia

Tabla 4.29

Caso de uso N°5: El jugador recolecta recompensas (Gemas)

Caso de uso N° 5

ID	CU05
Título	El jugador recolecta recompensas (Gemas)
Descripción	<ul style="list-style-type: none"> - El jugador selecciona e inicia un nivel - Durante el recorrido del nivel al jugador aparecen objetos flotantes en forma de gemas - El jugador al colisionar con cada uno de estos objetos, estas desaparecen y se contabilizan.

Referencias: 3

Condiciones de validación

- Se debe visualizar en la esquina superior izquierda el número de gemas recolectadas.

Elaboración propia

Tabla 4.30

Caso de uso N°6: El jugador pierde una vida al colisionar con un obstáculo

Caso de uso N° 6

ID	CU06
Título	El jugador pierde una vida al colisionar con un obstáculo
Descripción	<ul style="list-style-type: none"> - El jugador durante el recorrido puede correr o saltar - Cuando el jugador colisione con algún obstáculo inmediatamente pierde una vida. - El jugador regresa al último punto de control alcanzado.

Referencias: 3

Condiciones de validación

- Se debe contabilizar cada vez que el jugador pierde una vida
- No hay un límite de vidas perdidas
- Al finalizar el nivel el número de vidas perdidas debe adjuntarse al resumen del puntaje obtenido.

Elaboración propia

Tabla 4.31

Caso de uso N°7: El jugador desactiva un obstáculo (Trampa o compuerta)

Caso de uso N° 7

ID	CU07
Título	El jugador desactiva un obstáculo (Trampa o compuerta)
Descripción	<ul style="list-style-type: none"> - El jugador se encuentra con un obstáculo en el recorrido del nivel de juego - El jugador presiona la tecla “E” para acceder al panel de control del obstáculo - El jugador resuelve el puzle para desactivar el control - Se valida la solución y se cierra el panel de control - El obstáculo se desactiva

Referencias: 3

Condiciones de validación

- Se debe contabilizar el tiempo (Temporizador) que el jugador dispone para resolver el puzle.

- Al intentar desactivar el control del obstáculo y la solución no es correcta, se debe descontar 30s al temporizador.
- Al finalizar el proceso se debe deshabilitar el área de colisión del obstáculo así mismo se debe cambiar su estado y su animación.

Elaboración propia

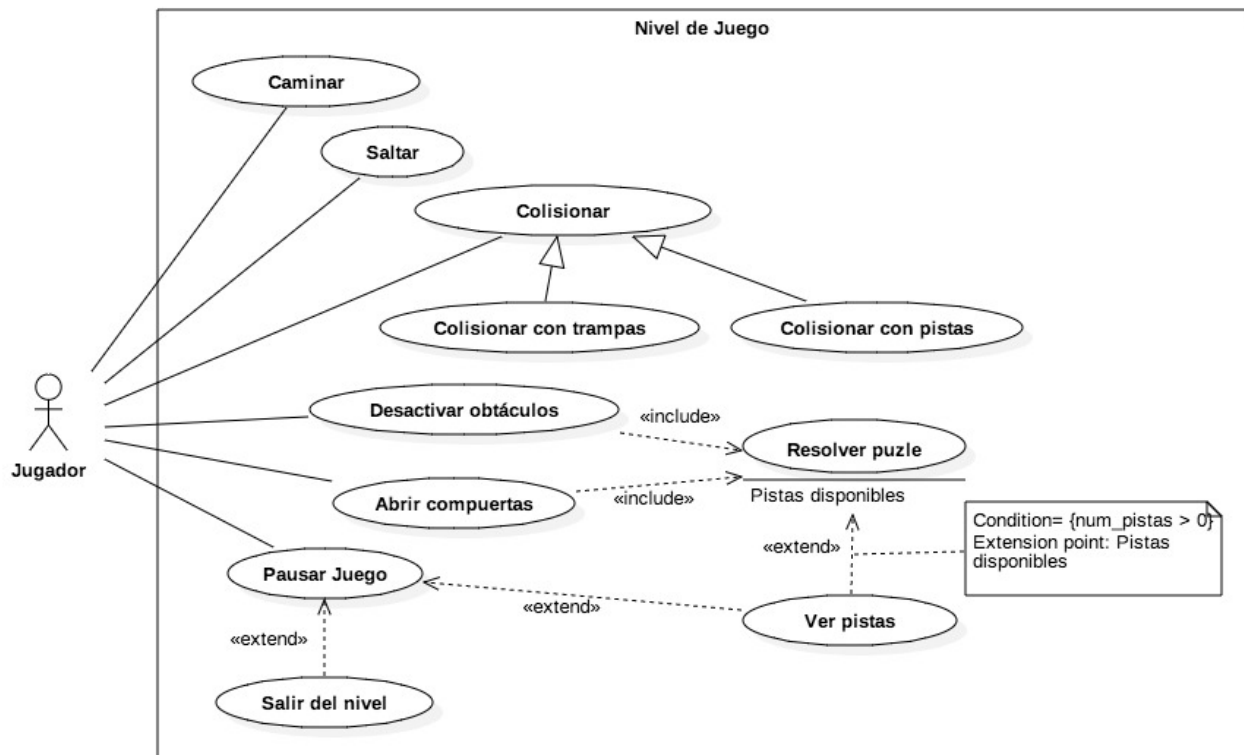


Figura 4.2 Vista de casos de uso Interacción del personaje con los objetos de juego
Fuente: Elaboración propia (2018)

Tabla 4.32

Caso de uso N°8: El jugador aplica una transformación geométrica

Caso de uso N° 8

ID	CU08
Título	El jugador aplica una transformación geométrica.
Descripción	<ul style="list-style-type: none"> - El jugador intenta resolver un puzle de Transformación Geométrica para desactivar un obstáculo. - Inicialmente se muestran las instrucciones (Indica el tipo de transformación y el número de píxeles que se deben modificar para resolver el puzle)

- El jugador selecciona del menú la opción (Aplicar transformación)
- Se muestran 3 opciones (Traslación, Rotación y Escalamiento)
- El jugador completa las matrices (Inicial, Transformación y Resultado) de acuerdo con las instrucciones.
- El jugador aplica la transformación.
- Se valida la solución del jugador.
- Se muestra visualmente el cambio en la matriz gráfica (píxeles)
- El jugador desactiva el obstáculo.

Referencias: 3

Condiciones de validación

- Se debe validar la sintaxis al momento de realizar las operaciones sobre la matriz principal (de la imagen) para que se pueda realizar la transformación correctamente.
- Cuando se evalúa la solución del jugador en caso de presentar algún error se debe devolver retroalimentación al jugador.
- Se debe almacenar los estados de la matriz principal para verificar un antes y después aplicada una transformación geométrica, este cambio debe visualizarse al finalizar el proceso.

Elaboración propia

Tabla 4.33

Caso de uso N°9: El jugador resuelve un algoritmo a través de TDL

Caso de uso N° 9

ID	CU09
Título	El jugador resuelve un algoritmo a través de un proceso TDL.
Descripción	<ul style="list-style-type: none"> - El jugador intenta resolver un puzle TDL para desactivar un obstáculo. - Inicialmente se muestran las instrucciones (Indica que para modificar la matriz principal se debe implementar el algoritmo de suma de matrices para el caso de traslación geométrica y multiplicación de matrices para el caso de escalamiento geométrico) dado que no está implementado como en los casos de uso anteriores. - El jugador selecciona del menú la opción (Resolver algoritmo) - Se muestra el marco de trabajo para crear casos de prueba.

- El jugador diseña los casos de prueba que el crea necesario.
- El sistema de validación revisa la sintaxis y evalúa el resultado de cada caso de prueba (suma o multiplicación de matrices).
- Validado los casos de prueba se muestra la mesa de trabajo para la implementación del algoritmo.
- El jugador implementa su solución.
- La solución del jugador se valida con los casos de prueba diseñados anteriormente.
- Si la solución es correcta se realizará la transformación geométrica con normalidad de lo contrario el jugador debe retomar la etapa de implementación.

Referencias: 3

Condiciones de validación

- Se debe validar la sintaxis y el resultado de los casos de prueba, en caso de que alguno de los ítems de las matrices sea incorrecto, se debe devolver retroalimentación de manera gráfica y textual al jugador.
- Al momento de evaluar el algoritmo implementado por el jugador, se deben mostrar todos casos de prueba se han superado y los que no.
- Al finalizar el proceso de la implementación de la solución se debe mostrar gráficamente el cambio en la matriz principal, en el caso de traslación se debe mostrar la reorganización de píxeles y en caso de escalamiento los píxeles que cambiaron de color.

Elaboración propia

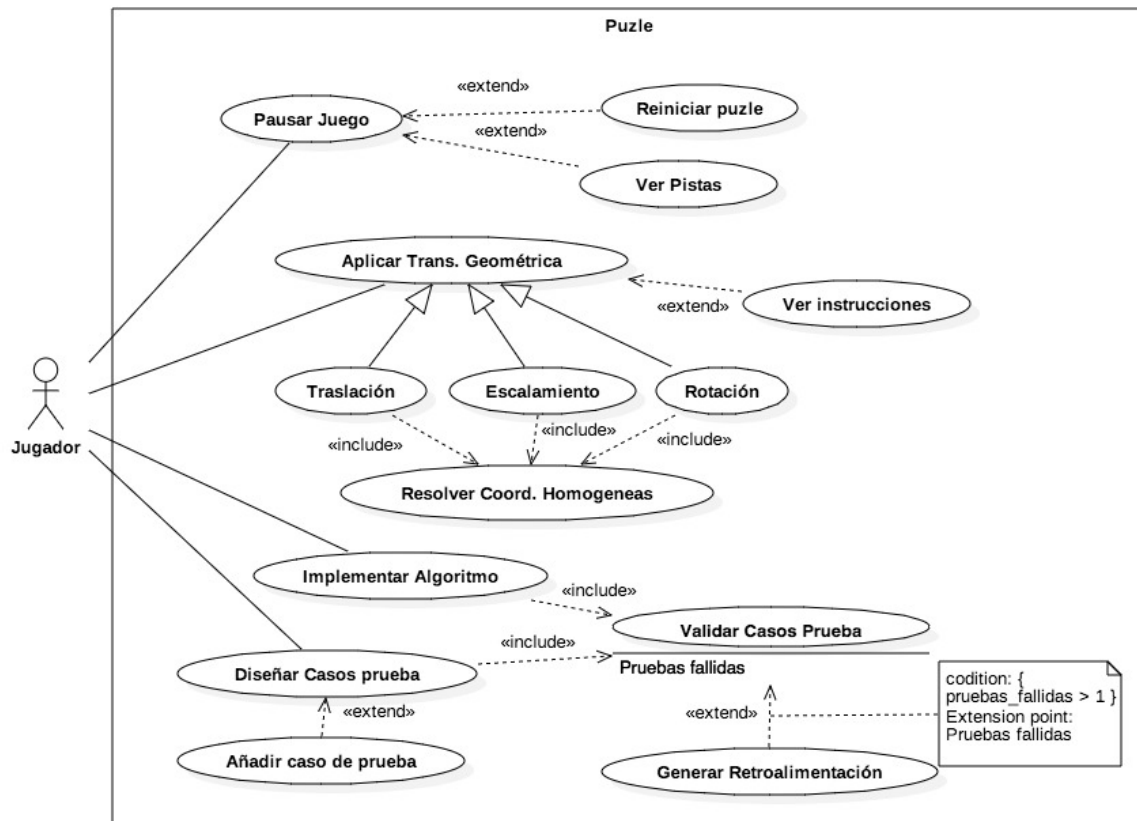


Figura 4.3 Vista de casos de uso para resolución de puzles de transformaciones y TDL

Fuente: Elaboración propia (2018)

2.2. Vista de procesos

Los niveles del juego se construyen en base a diferentes escenas, estas contienen los entornos donde se establecen los objetos, obstáculos entre otros artefactos para cada nivel del juego. Esta vista se enfoca en el diseño de escenas y su comportamiento en tiempo de ejecución, lo cual permite definir la continuidad del juego en relación con las acciones del jugador con la finalidad de modelar el rendimiento y comunicación [56]. A continuación, se presentan los diagramas de actividades de cada uno de los niveles o capítulos del Juego Serio en cuestión.

- a) **Nivel inicial:** Este nivel comprende las escenas que contienen la configuración inicial del juego y el menú de selección de niveles (Encargado de la continuidad del juego y la distribución de niveles).

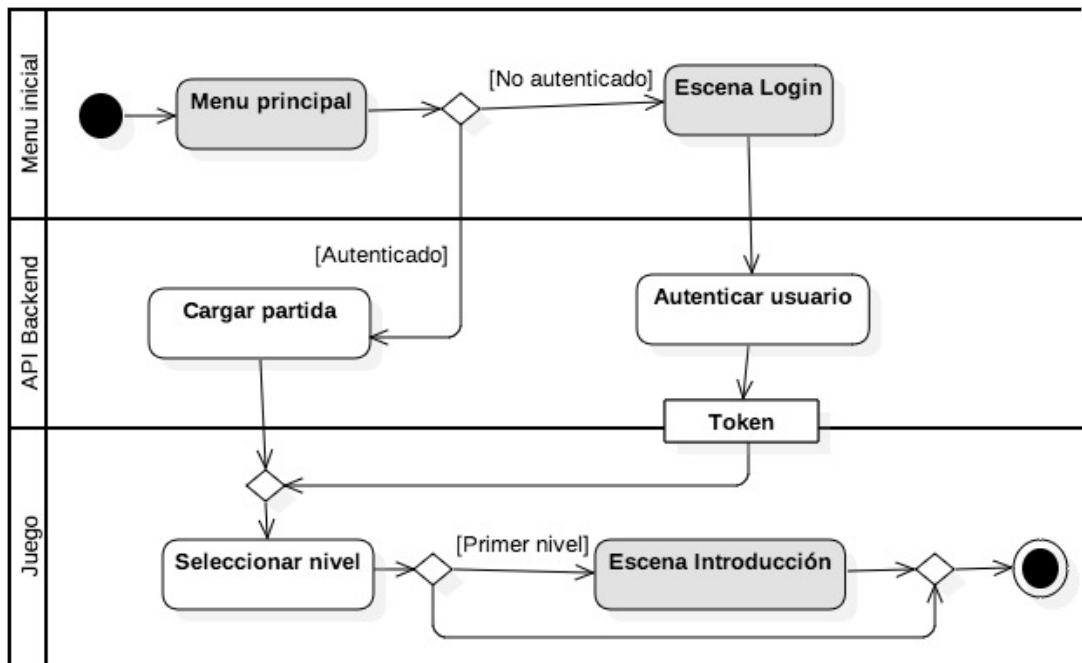


Figura 4.4 Diagrama de actividades escenas del nivel Inicial

Fuente: Elaboración propia (2018)

- b) **Nivel 1 (Capítulo I):** Este primer capítulo inicia en una escena principal (**Hall**) la cual se bifurca en 2 habitaciones contiguas (**Room A y Room B**), estas contienen obstáculos y compuertas que impiden el paso al jugador.

Cada obstáculo está conectado a un juego de lógica (Escena de puzle) el cual se debe resolver para avanzar en el recorrido del nivel. En este capítulo se encuentran los siguientes puzles: a) **Puzle Trivia** (Preguntas sobre matrices) b) **Puzle traslación I** (Problema de traslación geométrica) c) **Puzle traslación II** (Problema de traslación geométrica de dificultad media).

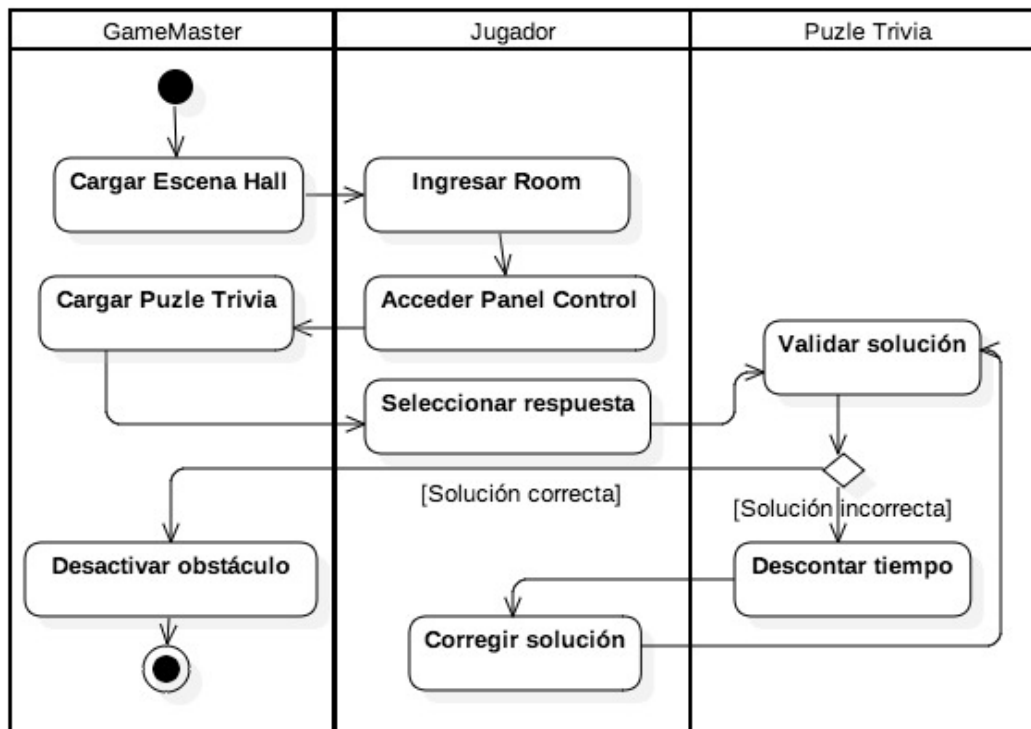


Figura 4.5 Diagrama de actividades de escenas del primer Nivel

Fuente: Elaboración propia (2018)

- c) **Nivel 2 (Capítulo II):** El segundo capítulo inicia en una escena similar al primer nivel (Hall 2do piso). Sin embargo, este escenario solo tiene una habitación (Room C), la cual continúa con la misma dinámica del capítulo anterior (resolver puzles para desactivar obstáculos y abrir compuertas).

En este nivel se encuentran los siguientes puzles a) **Puzle Traslación II** (Problema de traslación geométrica de dificultad media), b) **Puzle Rotación** (Problema de rotación geométrica) y c) **Puzle Escalamiento** (Problema de escalamiento geométrico)

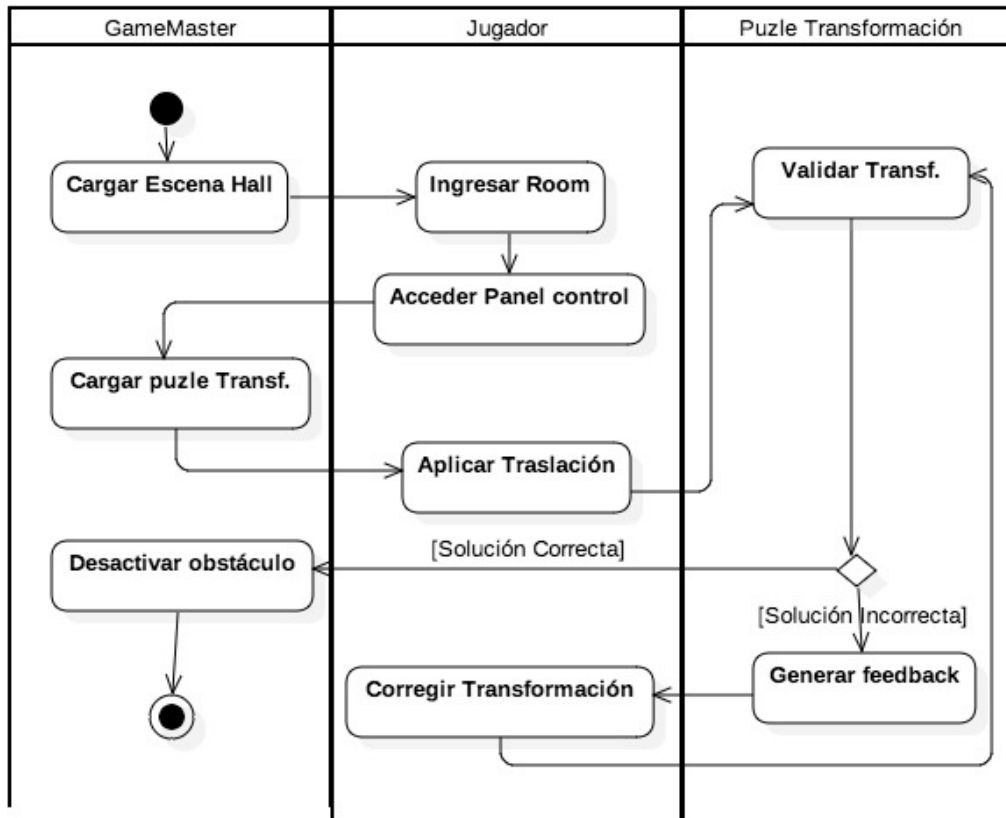


Figura 4.6 Diagrama de actividades de escenas del segundo Nivel

Fuente: Elaboración propia (2018)

- d) **Nivel 3 y 4 (Capítulo III y IV):** Estos últimos niveles mantiene la misma estructura y dinámica de los capítulos anteriores I y II. Los puzzles que se encuentran en este nivel se detallan a continuación: a) Puzzle de escalamiento (Problema de escalamiento geométrico), b) Puzzle TDL Suma (Implementación del algoritmo de suma de matrices basado en Test Driven Learning), y c) Puzzle TDL Multiplicación (Implementación del algoritmo de multiplicación de matrices basado en Test Driven Learning)

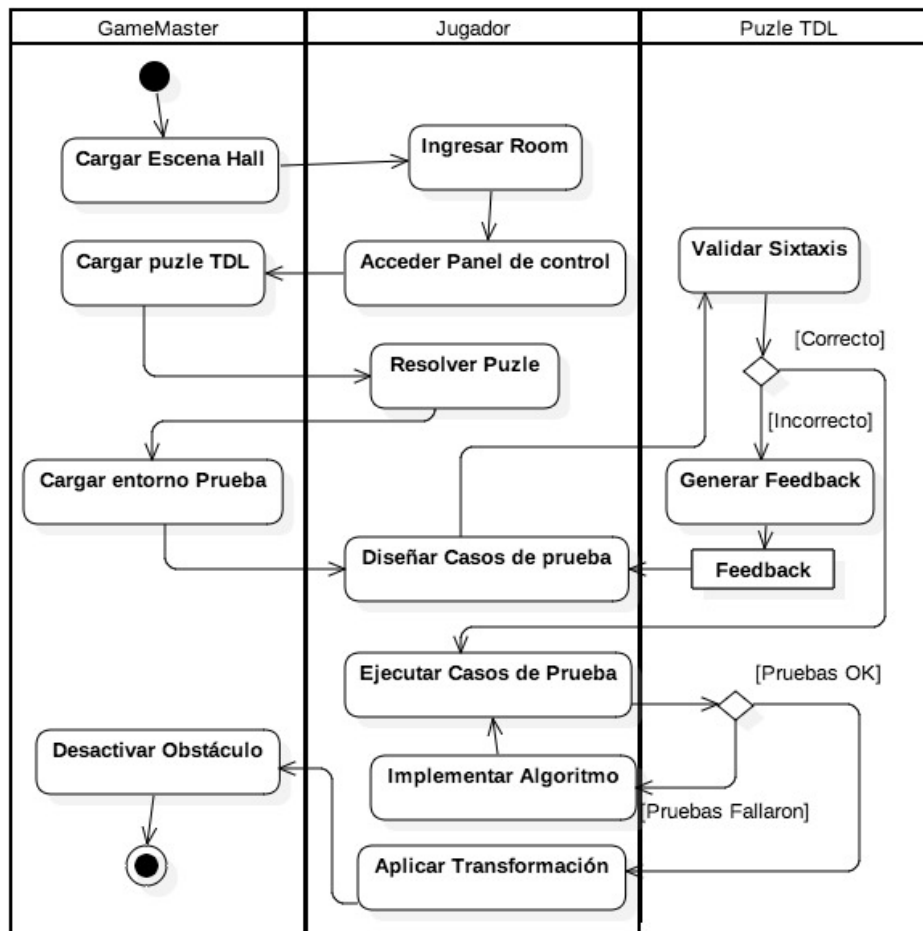


Figura 4.7 Diagrama de actividades de escenas del tercer Nivel

Fuente: Elaboración propia (2018)

2.3. Vista lógica

Esta vista de arquitectura describe la base para entender la estructura, organización y funcionalidad del Juego Serio [56]. A continuación, se detalla cada uno de los diagramas UML que comprende esta etapa (Clases y Ejecución) adicionando la vista lógica de datos (Entidad-Relación).

2.3.1. Diagrama de clases

La vista de clases permite trazar una estructura estática bajo el paradigma de programación orientada a objetos.

a) **Vista lógica Matrix Engine:** Esta vista describe las clases internas necesarias para la construcción de los puzles de transformación geométrica y implementación de algoritmos basados en TDL.

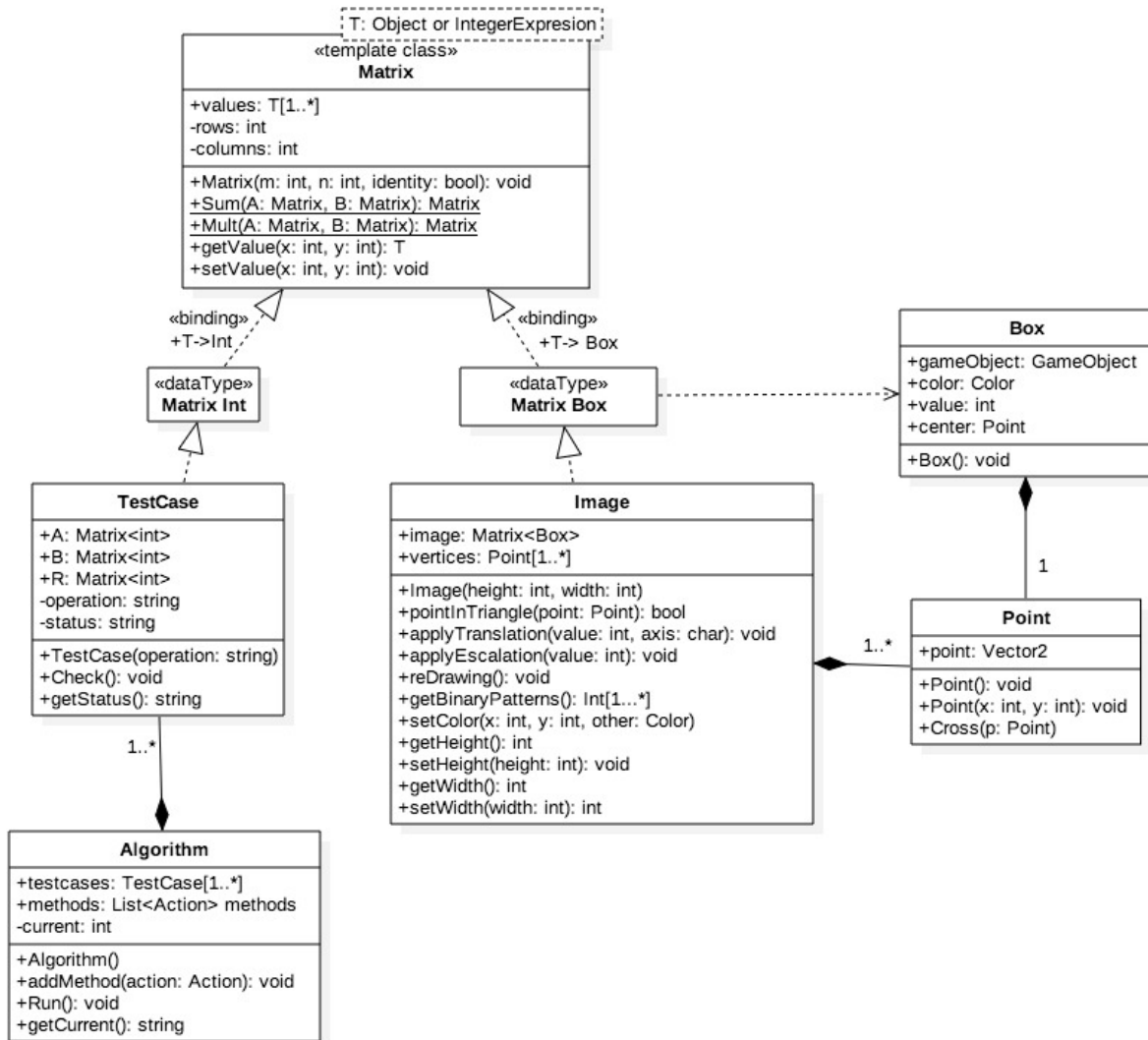


Figura 4.8 Diagrama de clases Matrix Engine & TDL

Fuente: Elaboración propia (2018)

Como se puede observar en el diagrama de la figura 4.8, la estructura se construye inicialmente sobre la clase base: **Matrix**, en la cual se declaran las operaciones básicas de matrices (Suma, Multiplicación, Igualdad).

Derivando la clase **Matrix** obtenemos la clase **Image**, la cual hereda las características y operaciones de la clase matriz, sin embargo, esta clase reemplaza

el contenido numérico heredado por un arreglo de dos dimensiones del tipo **Box** (píxel) con la finalidad de obtener una matriz de píxeles, en esta clase así mismo comprende los métodos necesarios para realizar transformaciones geométricas (Traslación, Rotación y Escalamiento). La clase **Generator** agrupa dos objetos del tipo *Image* (matriz gráfica inicial y la matriz solución) esta clase almacena la información y métodos necesarios para la resolución de puzzles de transformación geométrica.

Finalmente se modela la clase **ScratchMatrix** la cual también se deriva de la clase *Matrix*. Sin embargo, a diferencia de la clase *Image*, esta no almacena píxeles, sino campos editables del tipo *TextField*. Esta clase es utilizada en la creación de los puzzles TDL, específicamente para la creación de los casos de prueba (donde el jugador ingresa por teclado los valores de la matriz).

b) Vista lógica Controlador del Personaje

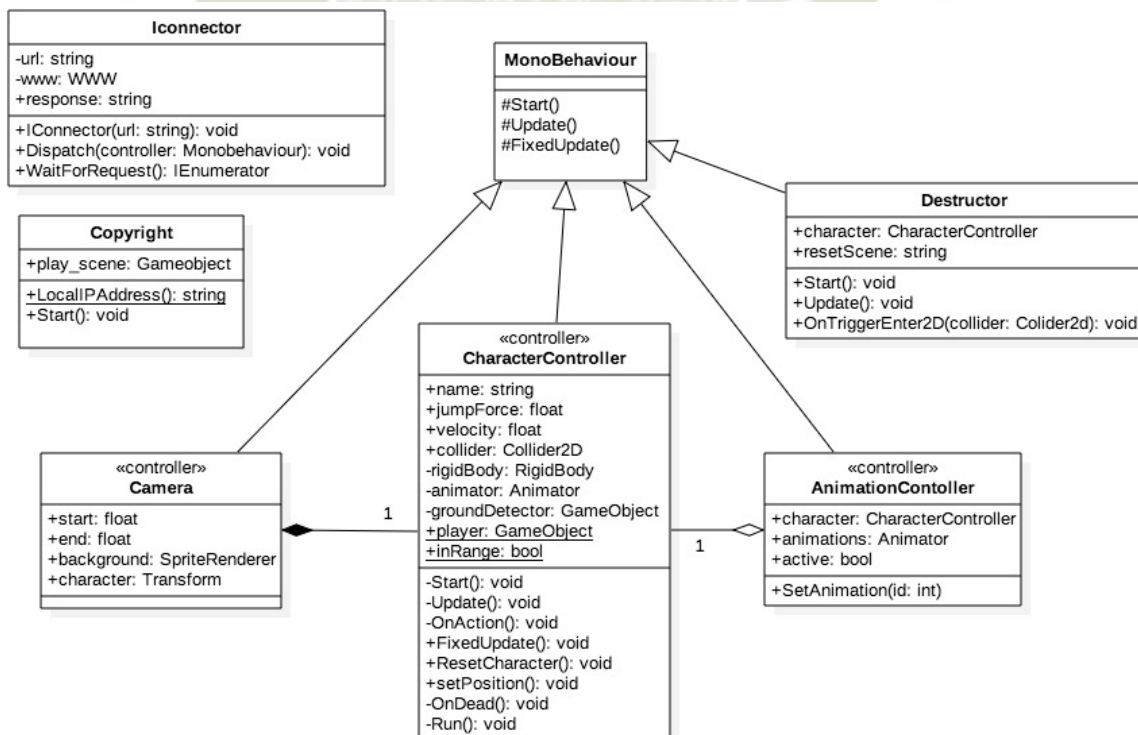


Figura 4.9 Diagrama de clases Controlador del personaje & Acceso a datos

Fuente: Elaboración propia (2018)

Esta vista describe mediante un diagrama de clases la estructura lógica del controlador del personaje, así como también los objetos que se relacionan indirectamente con la mecánica de juego.

El diagrama de clases que se muestra en la Figura 4.9 detalla algunas las clases internas del controlador del personaje. La clase principal (*CharacterController*) determina las acciones del personaje principal, esta clase instancia como parte de sus atributos, un objeto del tipo *Camera*, el cual permite el seguimiento visual del personaje en el recorrido de las escenas. Este objeto es esencial ya que sin el, no se podría visualizar ningún elemento gráfico en la pantalla.

Las clases *ActionController*, *DoorController*, *HintController* y *Destructor* utilizan una referencia de objeto del personaje (*CharacterController*), esto cual permite almacenar la ubicación en memoria del personaje principal, el cual se utiliza para lanzar eventos al momento colisionar con otro objeto de la escena. En el caso de la clase *Destructor* esta referencia se utiliza para actualizar la posición del personaje en el plano de la escena, a diferencia de los otros controladores que solo la utilizan en modo de lectura.

2.3.2. Diagrama de secuencia

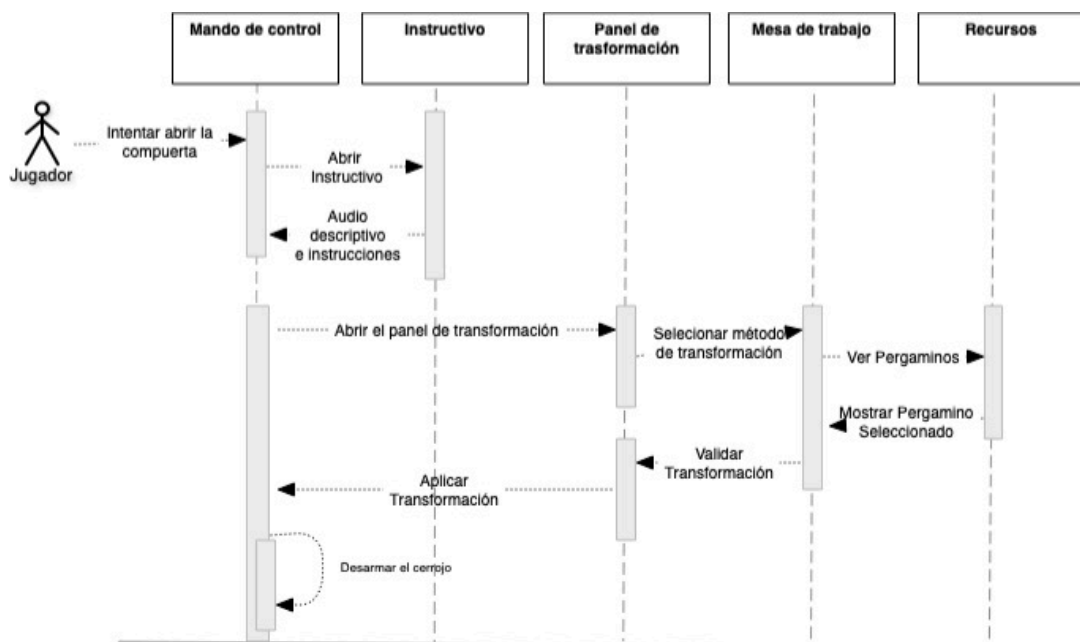


Figura 4.10 Diagrama de secuencia Resolución puzle de transformación

Fuente: Elaboración propia (2018)

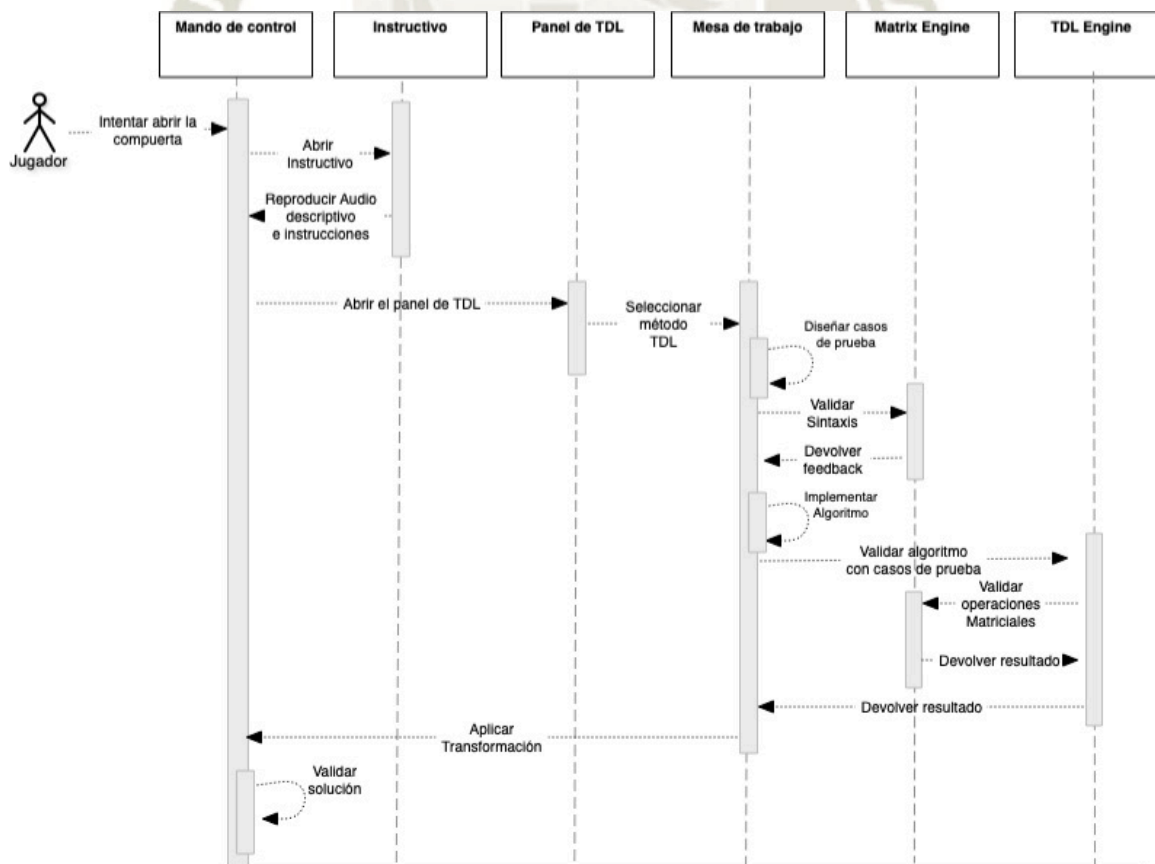


Figura 4.11 Diagrama de secuencia Resolución puzle TDL

Fuente: Elaboración propia (2018)

2.3.3. Diagrama Entidad Relación

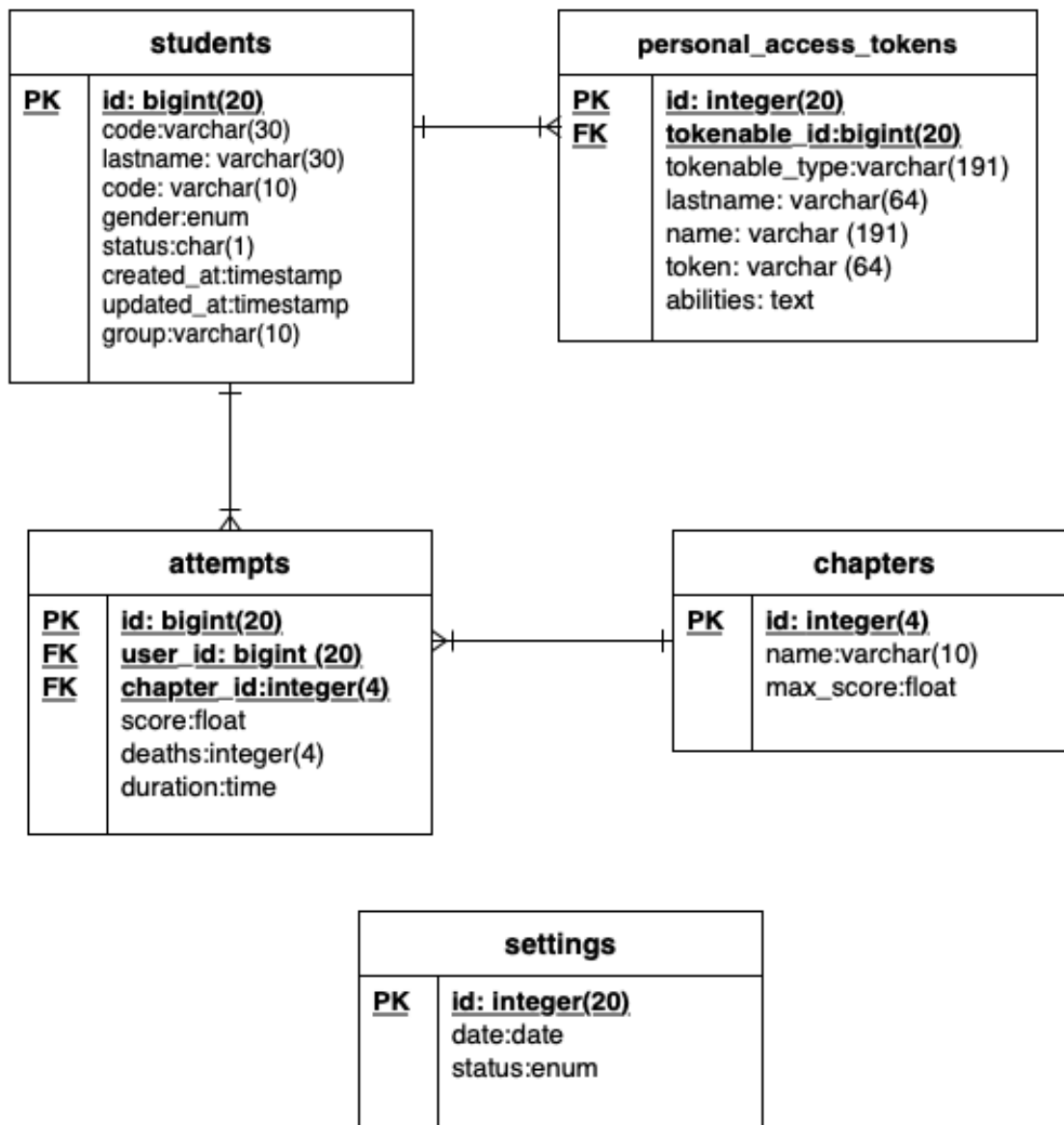


Figura 4.12 Diagrama Entidad Relación del Juego Serio

Fuente: Elaboración propia (2018)

2.4. Vista de despliegue (Desarrollo)

Esta vista describe la organización del Juego Serio desde una perspectiva de desarrollo [56].

Esta comprende los diagramas de componentes y de paquetes del software.

2.4.1. Diagrama de componentes

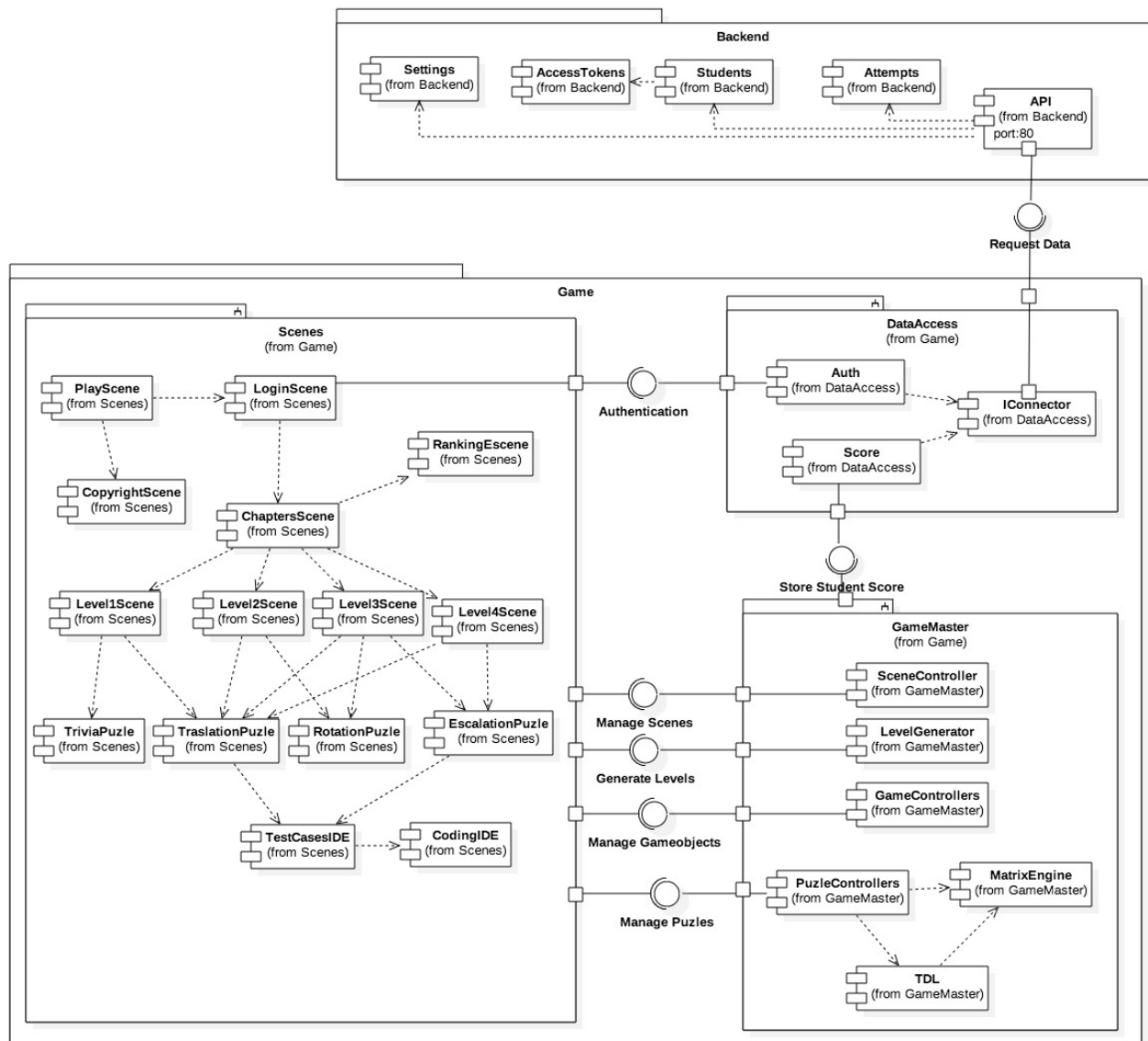


Figura 4.13 Diagrama de componentes del Juego Serio

Fuente: Elaboración propia (2018)

2.4.2. Diagrama de paquetes

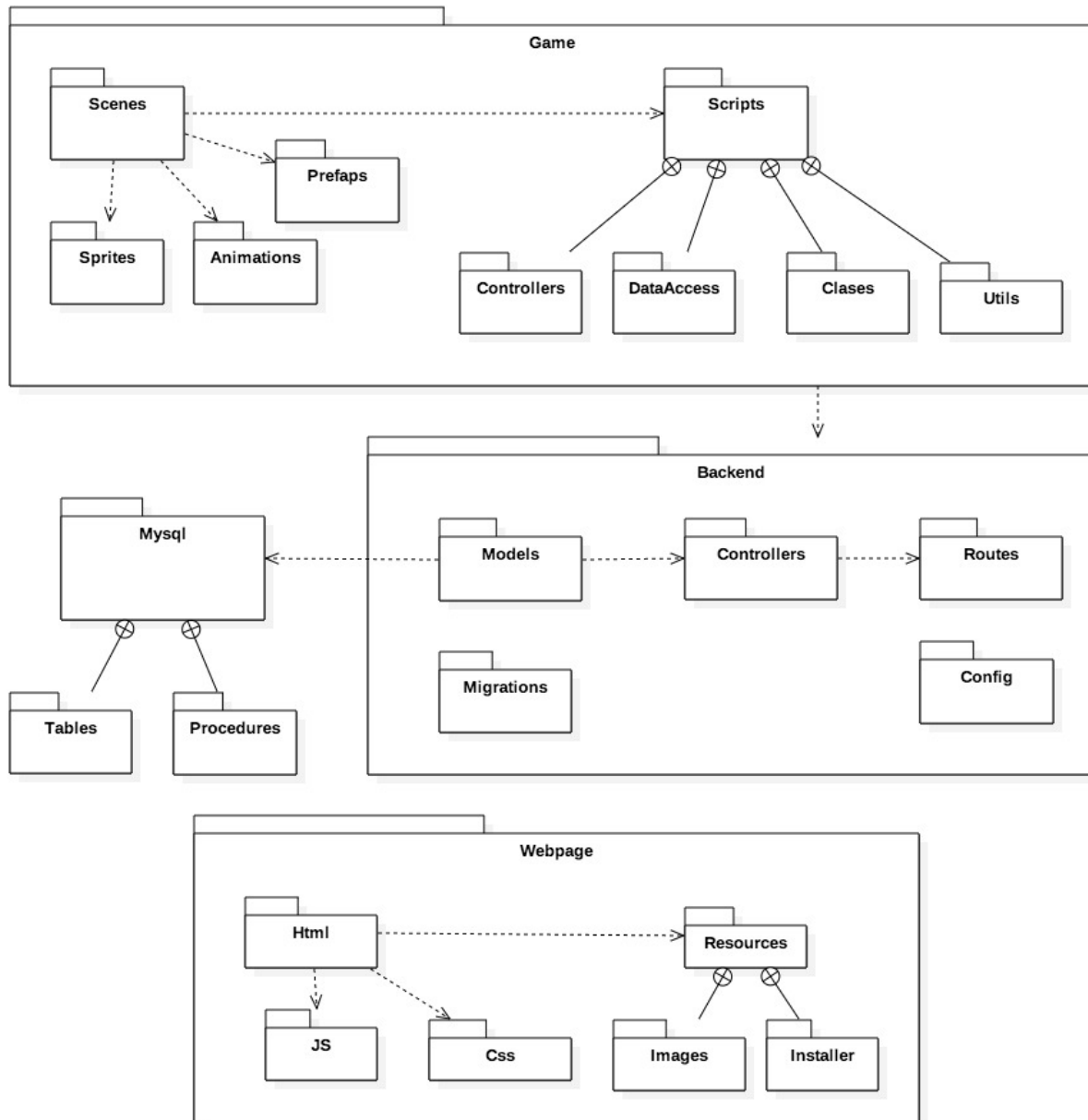


Figura 4.14 Diagrama de paquetes del Juego Serio

Fuente: Elaboración propia (2018)

2.5. Vista física

La vista física describe la topología del sistema desde un punto de vista general que refleja la distribución de los componentes del software [56].

2.5.1. Diagrama de despliegue

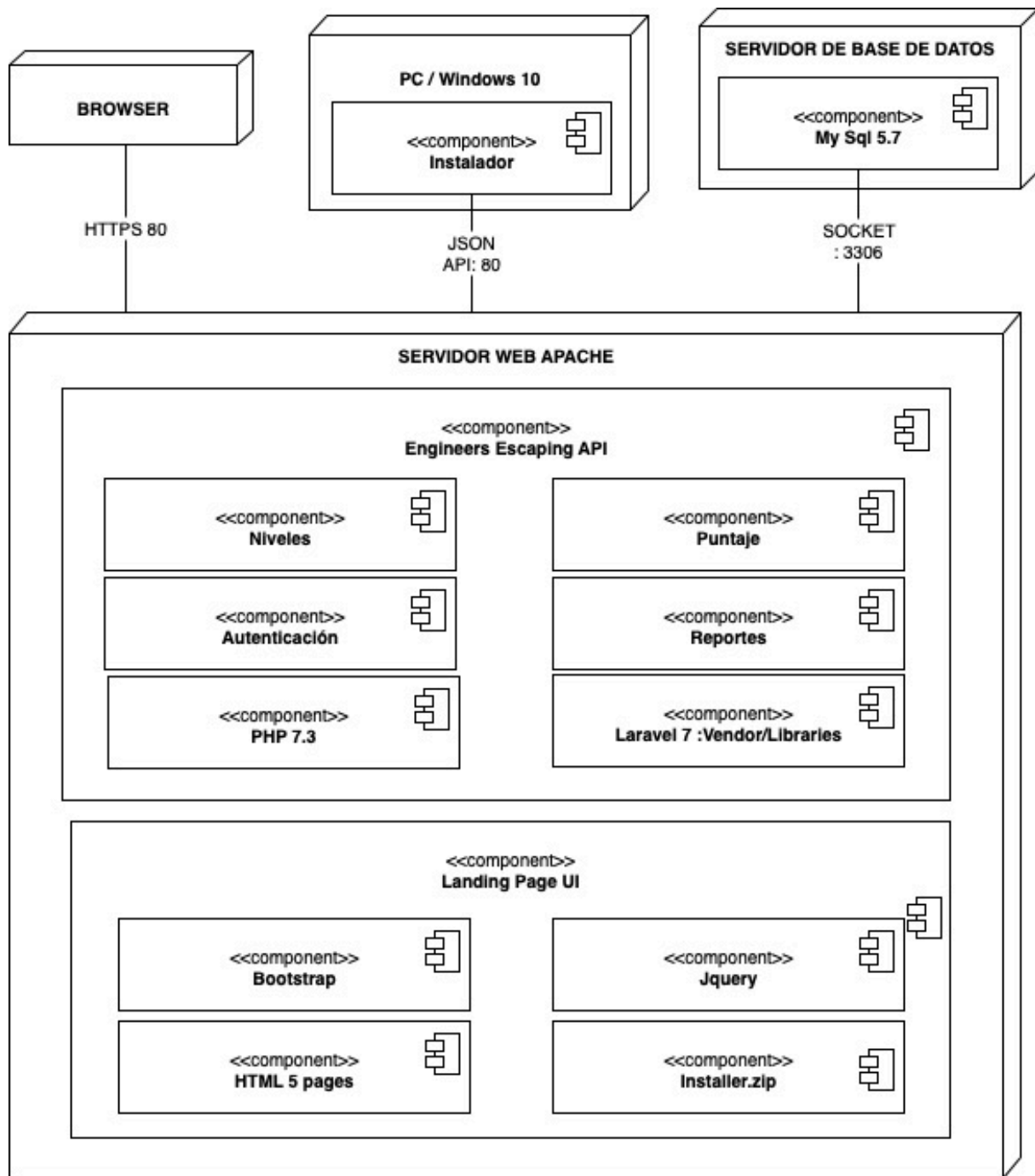


Figura 4.15 Diagrama de despliegue del Juego Serio

Fuente: Elaboración propia (2018)

3. Conceptualización (Preproducción)

3.1. Nombre e iconografía del juego

El título e iconografía del Juego Serio propuesto se definió en relación con la narrativa propuesta, así mismo se utilizó al personaje principal en la creación del icono final utilizando como referencia las de uso comercial.



Figura 4.16 Diseño de icono del Juego Serio

Fuente: Elaboración propia (2018)

3.2. Narrativa del juego

La historia como menciona [36] debe representar una situación real donde el jugador se ve inmerso, es por ello por lo que se optó por tomar una actividad cotidiana del estudiante como el hecho de asistir a clases en la universidad. Por lo tanto, nuestra propuesta (Engineers Escaping) ejemplifica una situación real en la vida cotidiana de un estudiante de ingeniería de sistemas.

Engineers Escaping es un videojuego que ejemplifica una situación real en la vida cotidiana de un estudiante de ingeniería de sistemas. La historia se centra en el salón de clases de un curso de programación, el protagonista descubre una cinta que le indica que los ingenieros han sido secuestrados y deberá rescatarlos, ya que dependerá su nota del curso hacerlo en el menor tiempo posible. Ver anexo C. Guía de juego.

3.2.1. Características principales:

- a) **Planteamiento sencillo:** La historia es simple, suficientemente explícita para que los estudiantes entiendan el objetivo.
- b) **Táctica:** El juego es del tipo plataforma, las acciones comunes son moverse, saltar y evitar obstáculos para llegar a puntos de control o “Checkpoints” que guardarán su progreso. La habilidad requerida por parte del jugador son el uso de los comandos básicos (Direccionales y Barra espaciadora) para realizar estas acciones.
- c) **Dinamismo:** Nuestra propuesta plantea un estilo de suspenso con la finalidad de provocar una sensación de tensión en el jugador y promover su concentración durante el tiempo de juego.
- d) **Escalabilidad:** El Juego Serio puede generar niveles de manera aleatoria de acuerdo con su configuración global, esto permite generar escenarios variados y que el jugador pueda volver a jugar siempre que lo desee.

3.2.2. Género

El Juego Serio en cuestión se desarrolló en base a la unión de dos géneros (Plataformas: Escape Rooms con Juegos de trivia o lógica). A continuación, se describen cada uno de ellos:

- a) **Escape Rooms:** Subgénero de videojuegos enfocados principalmente a generar tensión al jugador, pretenden provocar interés, estado de concentración en el jugador [57]. Este tipo de juegos, se encuadran dentro del género de plataforma (Aventura/ Exploración). Para nuestra propuesta: “Engineers Escaping” los objetos como trampas y escenarios fueron desarrollados bajo estas características.

b) Trivia o lógica: También conocidos como juegos de puzzle, género de videojuegos que se caracteriza por poner a prueba la lógica y sus conocimientos del jugador. Por lo general este tipo de juegos suele no presentar una identidad única, debido a que esta guarda relación directa con la mecánica del juego de un Escape Room [58].

3.2.3. Mecánica

En esta sección entraremos más en detalle en lo que a las mecánicas de Engineers Escaping se refiere. Se detallarán todas las características que fundamentan su jugabilidad, así como las acciones que llevará a cabo el jugador dentro de una partida. Además, se detalla una lista con los personajes del juego (tanto protagonista como trampas), habilidades, objetos etc. Por último, se modelará las escenas en el plano de movimiento, físicas y detección de colisiones.

3.2.4. Dinámica

El juego es de tipo Plataformas (juego que se caracteriza por contener bloques y estratagemas, donde el jugador puede saltar, correr y atravesar estos componentes) En este tipo de juegos la narrativa no es tan evidente, existe una historia, objetivo o misión y se hace énfasis en la habilidad del jugador para avanzar en el juego.

3.3. Diseño de interfaces y prototipos

Durante la etapa de planificación surge abundante incertidumbre sobre la iniciación del proyecto, en una primera labor es diseñar los componentes que forman parte del juego (personaje principal, obstáculos, recompensas, diálogos, etc) [48].

3.3.1. Diseño del concepto

Para realizar el diseño del concepto del juego es necesario bosquejar los posibles escenarios en que este podría darse, esto quiere decir que se diseña a mano alzada los **storyboards** donde interactúa el jugador con los otros elementos del juego este proceso ayuda a centralizar en concreto los objetivos del juego.

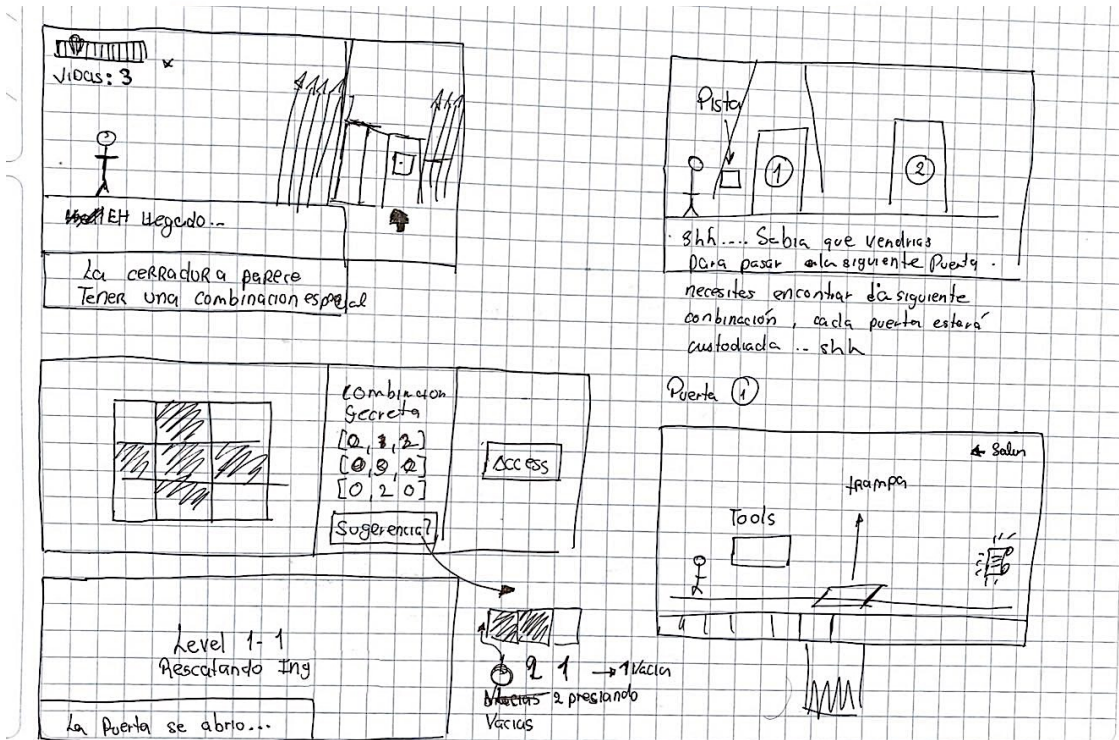


Figura 4.17 Diseño del concepto del primer nivel

Fuente: Elaboración propia (2018)

Como se observa en la Figura 4.17 también se incluyen los posibles diálogos que se le presentan al jugador, entre algunos otros componentes y funcionalidades. Sin embargo, algunas características fueron evolucionando para cumplir con los objetivos del proyecto como lo fueron los puzzles en cada nivel y la manera como irían apareciendo y superando por el jugador. Un ejemplo de esto se puede observar en Figura 4.18, en una primera instancia se propuso el uso de llaves para desbloquear los controles, para obtener estas llaves era necesario resolver los

puzles. Sin embargo, se asumió una ineficacia en su uso, ya que los controles se mostrarían como una caja negra para el estudiante ignorando el contenido didáctico.

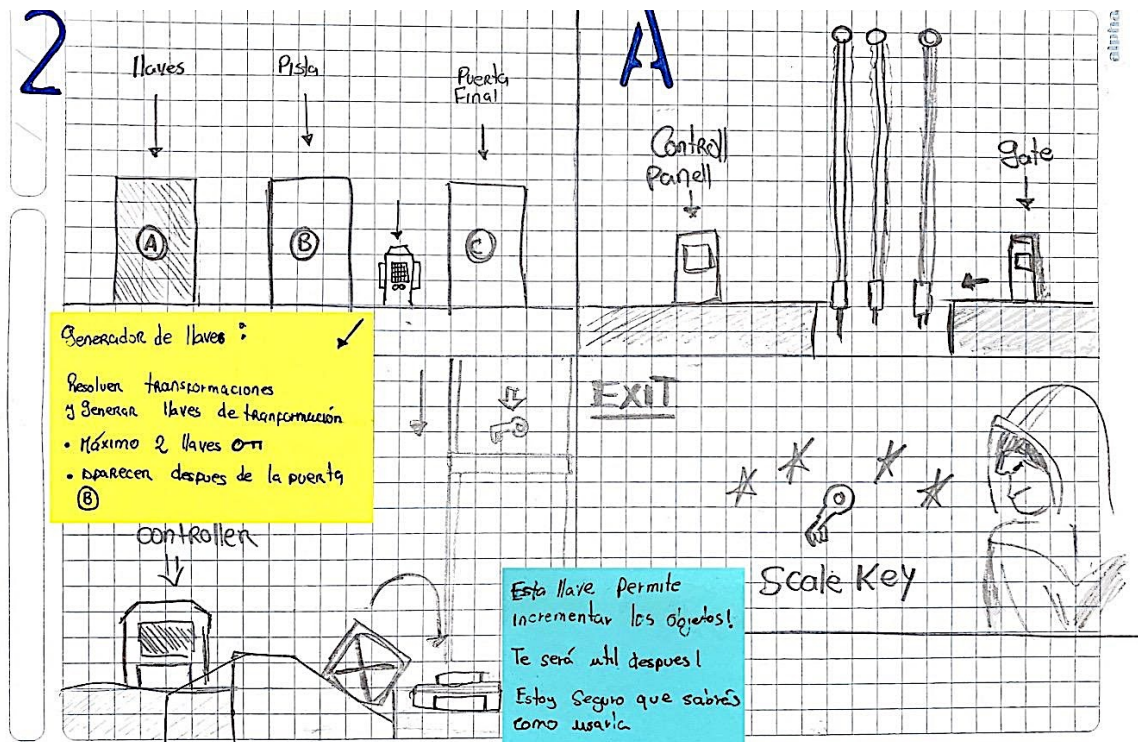


Figura 4.18 Diseño del concepto 2do nivel (Capítulo II)

Fuente: Elaboración propia (2018)

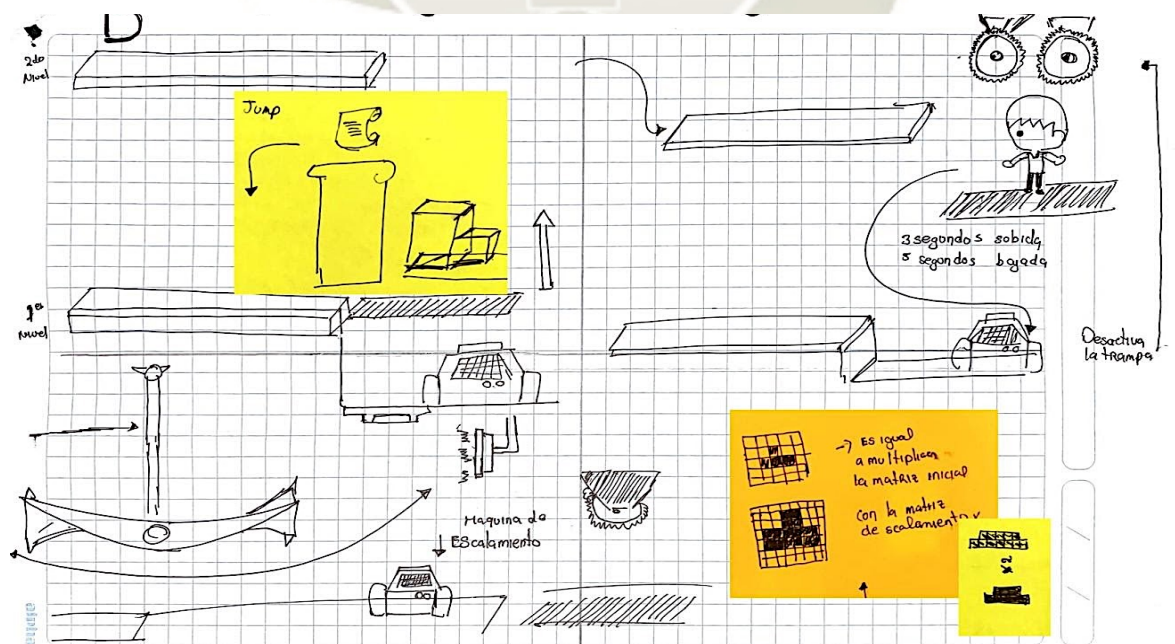


Figura 4.19 Diseño de concepto 3er nivel (Capítulo III)

Fuente: Elaboración propia (2018)

3.3.2. Personajes y elementos del juego

La primera tarea fue crear al personaje principal, realizar algunos bosquejos y sus posibles acciones dentro del juego, posteriormente en base a ello construir una historia que proporcione al jugador pertenencia e identidad con el storytelling de sucesos.

a) Protagonista

Es el personaje principal que representa al jugador, es el único elemento controlable en el juego. El personaje comprende todas las acciones que el jugador puede tomar a lo largo de toda la historia del juego.

- **Habilidades:** Saltar, caminar, activar controles y activar puntos de control (Checkpoints), recolectar pistas y recompensas en el juego.
- **Animaciones:** De acuerdo con las habilidades descritas en el punto anterior, las animaciones proyectadas son las de caminar, saltar y al momento de colisionar con un obstáculo (Perder una vida).
- **Diseño:** De acuerdo con las acciones del personaje principal el bosquejo es el siguiente:

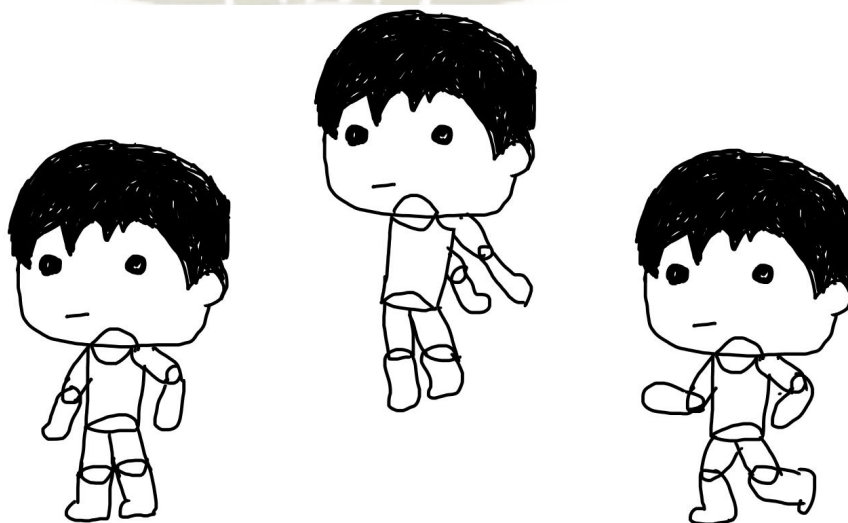


Figura 4.20 Diseño a mano alzada de las acciones del protagonista

Fuente: Elaboración propia (2018)

Como se puede observar en la Figura 4.20, el personaje está compuesto de sub-imágenes que representan cada parte de su cuerpo, esto permite una independencia entre ellas y facilita la animación del personaje en la etapa de producción. A continuación, se presenta el diseño a mano alzada del Spritesheet para el personaje principal (L=Izquierda, R=Derecha, M=Torso, Sh=Vestimenta, H=Cabeza, HR=Cabello)

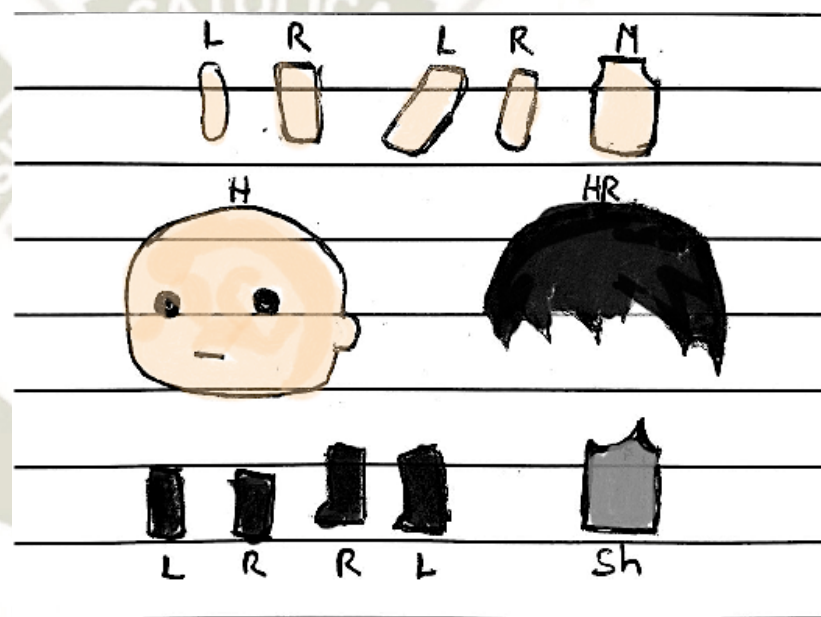


Figura 4.21 Diseño del Sprite-sheet del personaje principal

Fuente: Elaboración propia (2018)

b) Trampas y obstáculos

Son todos aquellos objetos que detienen el paso al jugador en el recorrido del juego. Estos objetos mantienen un estado (Encendido y apagado), tienen asociado un área de colisión con el personaje principal, esto quiere decir que su única función es provocar que el jugador pierda una vida.

La mayoría de estos se pueden evitar a través de las estrategias del jugador, sin embargo, otros requieren de la resolución de un puzzle para poder

desactivarlos. A continuación, se detalla el diseño a mano alzada de cada uno de los obstáculos que aparecen en el juego.

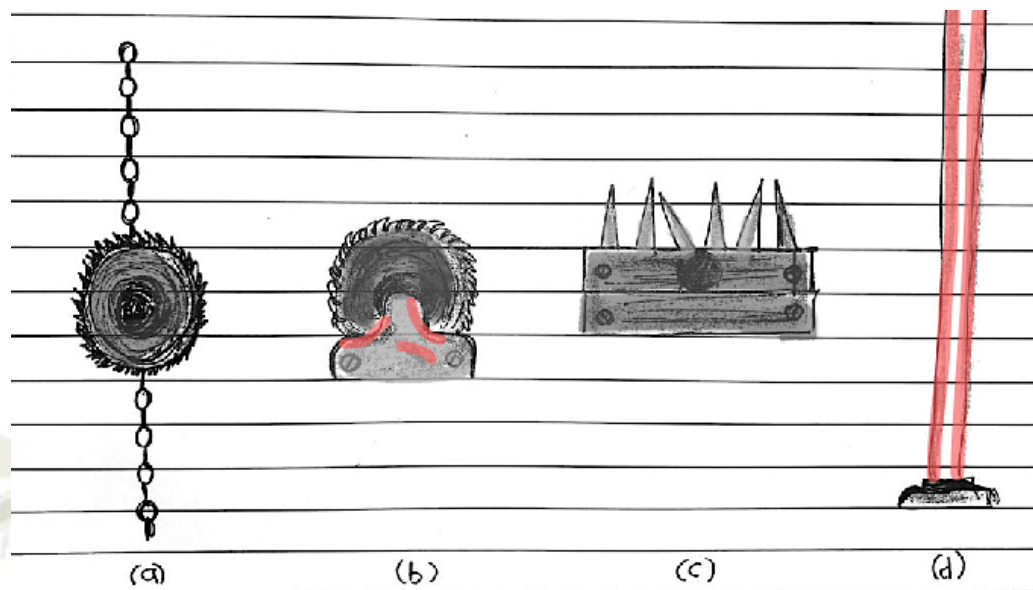


Figura 4.22 Diseño a mano alzada de los obstáculos

Fuente: Elaboración propia (2018)

- **Sierras Eléctricas**

Estas trampas presentan un área de colisión circular, obstaculizan al jugador para llegar a los “**Checkpoints**” (Donde se graba el progreso del jugador), la mayoría de estas pueden ser evitadas con las acciones del jugador (saltando sobre ellas, por ejemplo) sin embargo otras solo pueden desactivarse resolviendo los ejercicios de lógica (puzle de matrices).

- **Plataformas con púas**

Estas trampas solo obstaculizan el paso al jugador, para evitarlas el jugador debe esperar a que las púas descendan para poder cruzar. Estos objetos solo obstaculizan el paso, por lo tanto, no tienen asociado ningún juego de lógica para superarlos.

- **Puerta láser**

Esta trampa presenta un área de colisión rectangular vertical, por lo que si el jugador tiene contacto con este ocasiona la muerte al personaje principal. Este obstáculo contiene un ejercicio de lógica (puzle de matrices) que el jugador debe resolver para superarlos.

- **Trampa colgante**

Este obstáculo tiene un área de colisión de gran tamaño, y por lo general evita que el jugador pueda saltar, obligando así a caminar sobre las plataformas del juego. Al producirse una colisión con este objeto produciría la muerte del protagonista.

c) Panel de control

Este objeto está asociado a un obstáculo, presenta un área de colisión inofensiva que permiten al jugador acceder al ejercicio de lógica (puzle interno). El mecanismo de este objeto consta de solo acercarse a él y presionar la tecla “E”, al momento de detectar ese evento el jugador es re-direccionado a un puzle de lógica.

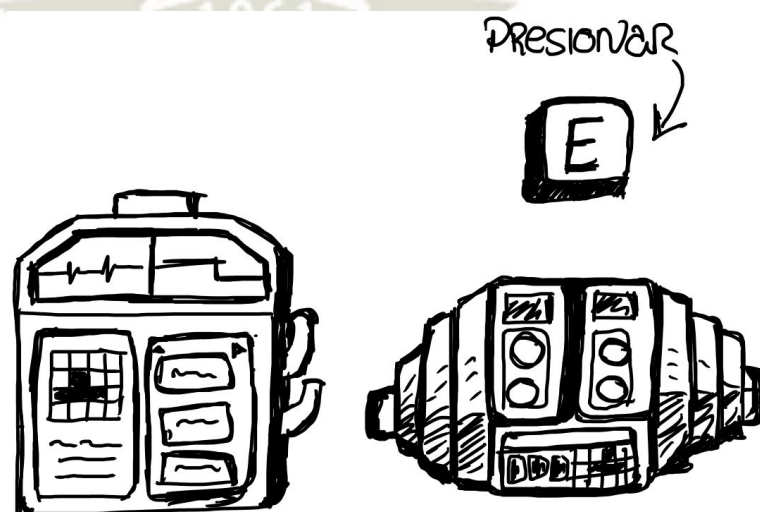


Figura 4.23 Diseño a mano alzada de los paneles de control

Fuente: Elaboración propia (2018)

Interacción entre el jugador y los elementos de juego

Bajo el contexto que se presentó anteriormente, el jugador está constantemente interactuando con otros elementos del juego: obstáculos, paneles de control, puertas, pistas, recompensas entre otros. Sin embargo, cada uno de estos tiene un impacto diferente sobre el personaje principal. En definitiva, las colisiones y los efectos que se producen son los siguientes:

Tabla 4.4.34 Colisión - Efecto del personaje principal

Objeto	Tipo	Efecto
Puerta	Acceso	Redirecciona al jugador a un nuevo escenario
Sierra Eléctrica / Puerta Laser / Plataformas con púas / Trampa colgante	Obstáculo	Ocasiona la muerte del personaje
Panel de control	Acceso	Redirecciona al jugador a un puzle de lógica
Gemas	Recompensa	Incrementa el contador de gemas
Pista	Recompensa	Añade una pista a los recursos del jugador

Elaboración propia

3.3.3. Wireframes del juego

Una vez culminado el diseño de concepto del juego, se realizan los “Wireframes” o planos de pantalla, vistas que se utilizan como guía en la etapa de producción o de desarrollo del juego, lo que permite realizar una correcta distribución de los elementos del juego sobre un escenario determinado. Fue realizado utilizando el programa *Runway*. A continuación, se presentan cada una de ellas:

a) Menú principal

Vista inicial del juego, contiene las acciones principales del juego (Jugar, Ver créditos).

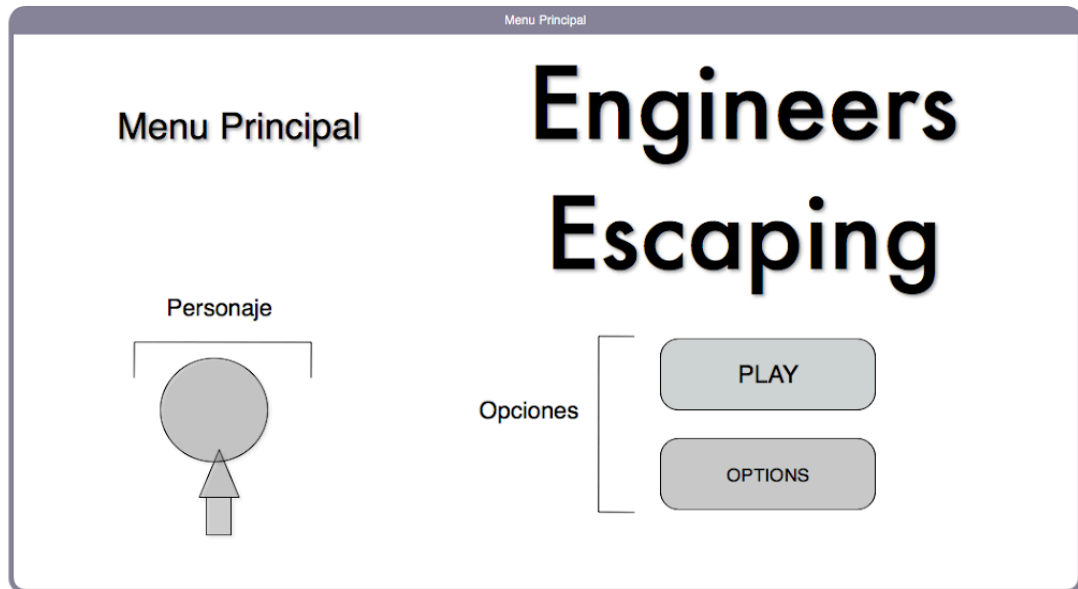


Figura 4.24 Wireframe: Menú principal

Fuente: Elaboración propia (2018)

- **Botón “Jugar”:** Si el jugador es nuevo al pulsarlo desencadena una escena de introductoria donde se relata la historia del juego, de lo contrario el jugador es redirigido a la escena de capítulos.
- **Botón “Créditos”:** Al pulsarlo lleva al jugador a los créditos del juego.
- **Botón “Salir”:** Al pulsarlo cierra el juego.

b) Menú de niveles (Capítulos)

Vista donde se presentan todos los niveles disponibles del juego, estos irán apareciendo conforme al progreso del jugador. Además, esta ventana relaciona cada nivel con su ranking (Listado de puntajes).

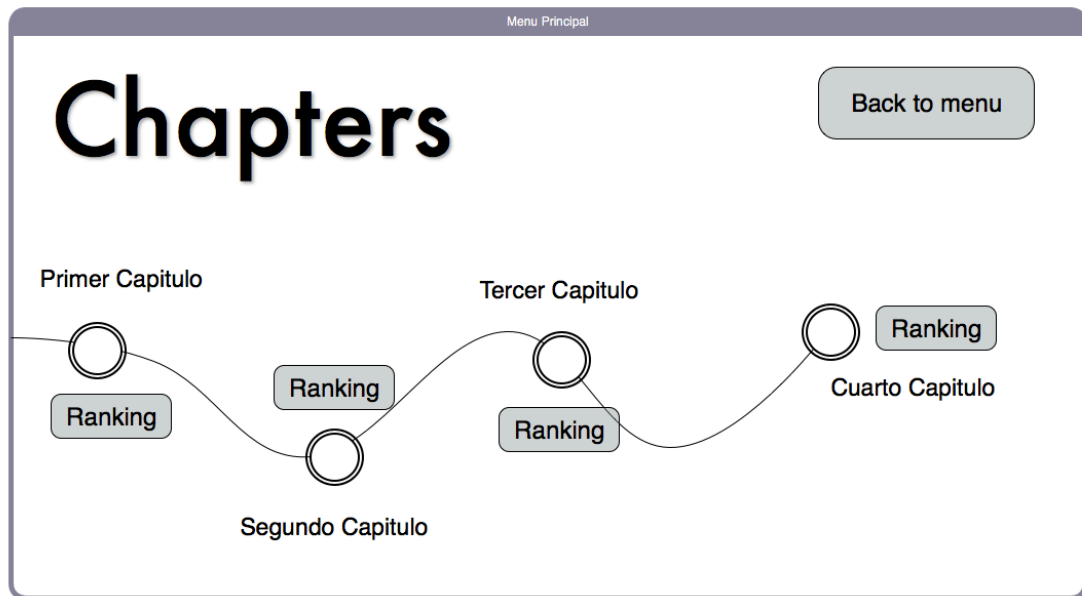


Figura 4.25 Wireframe: Menú de niveles (Capítulos)

Fuente: Elaboración propia (2018)

- **Capítulo [1,2,3,4]:** Al pulsarlo lleva a la escena del capítulo, siempre y cuando el jugador lo haya jugado anteriormente.
- **Botón “Back to Menu”:** Al pulsarlo regresa al jugador al Menú Principal.
- **Botón “Ranking”:** Al pulsarlo lleva al jugador a la pantalla de Ranking donde podrá visualizar su puntaje en contraste con el resto de los jugadores.

c) Tutorial

Esta vista presenta el tutorial del juego (El GamePlay y los controles básicos del teclado). Así también, una breve introducción al objetivo del juego (Rescatar al ingeniero secuestrado) y a las acciones que el jugador puede tomar durante el recorrido de cada uno de los niveles.

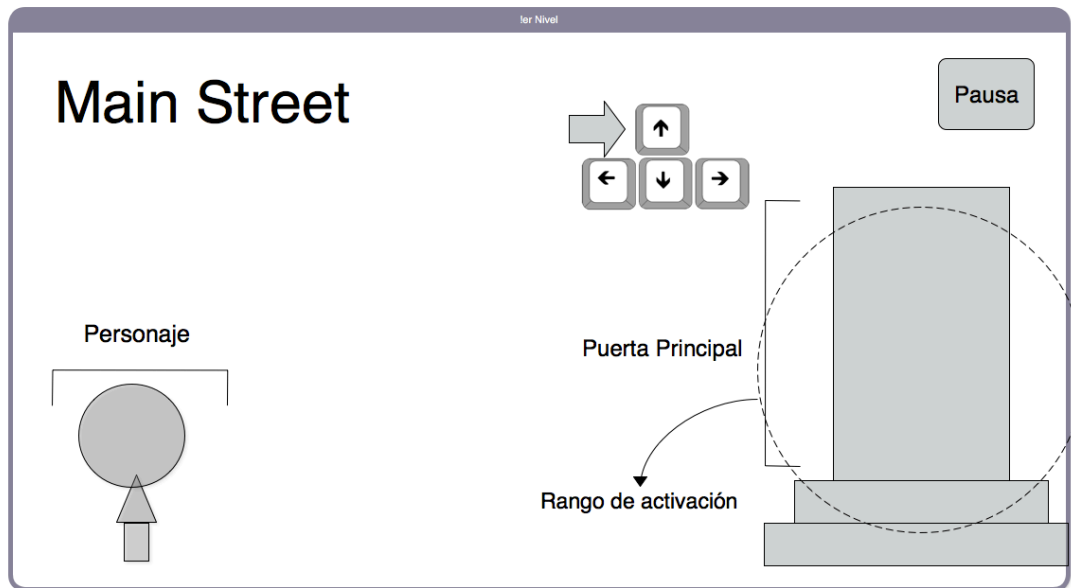


Figura 4.26 Wireframe: Nivel de introducción al juego

Fuente: Elaboración propia (2018)

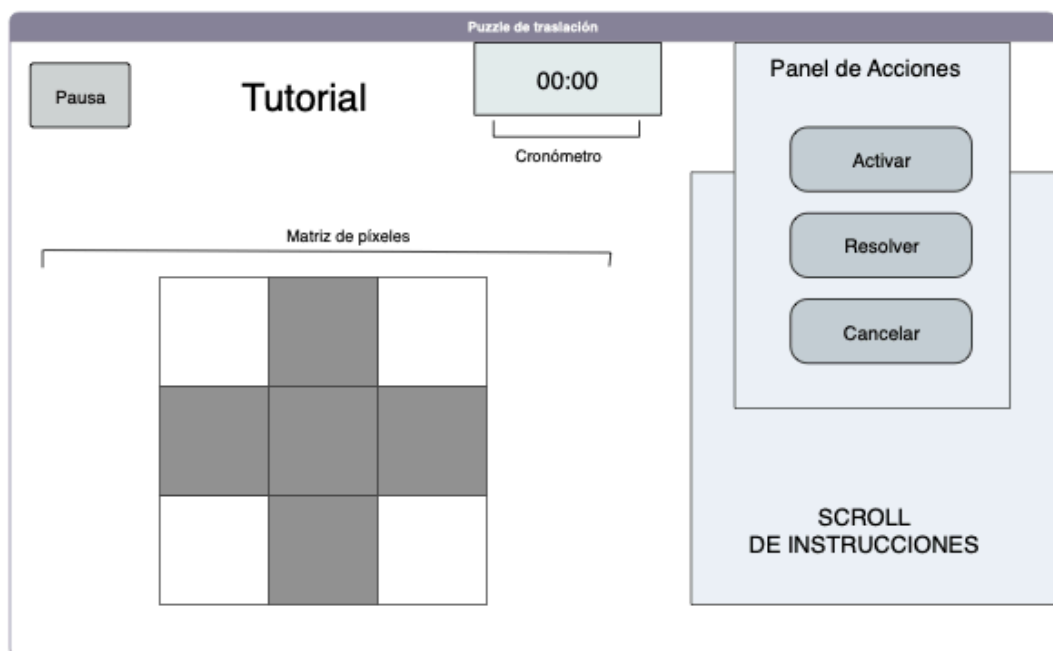


Figura 4.27 Wireframe: Puzle de introducción

Fuente: Elaboración propia (2018)

Cuando el jugador se encuentra en el rango de activación de la puerta principal, este debe pulsar Arriba (↑) para ingresar. En ese momento, se le llevará al jugador al puzle tutorial donde se le explica brevemente cada una de las acciones que se puede realizar.

- **Botón “Pausa”:** Al pulsarlo lleva al menú de pausa, donde el jugador puede regresar al Menú Principal.
- **Píxel:** Al hacer clic sobre cualquier píxel de la matriz gráfica este cambiará de color de blanco a negro y viceversa.

d) Recepción (Hall)

Siempre que el jugador inicie un nivel, este comenzará desde una escena principal denominada “Hall” la cual bifurca a las otras del nivel.

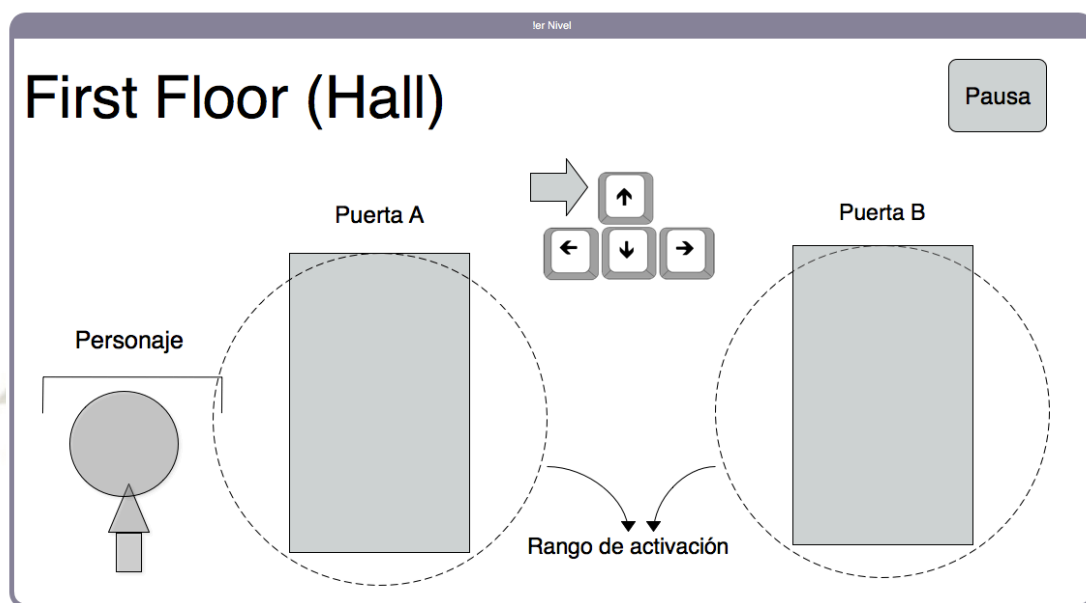


Figura 4.28 Wireframe: Recepción 1er Nivel (Hall)

Fuente: Elaboración propia (2018)

e) Primer nivel

Al inicio del primer nivel (ingresando en alguna de las puertas de la vista anterior), el jugador dará con el primer obstáculo y panel de control del juego. Este último lo re-direccionará al juego de lógica (puzzle de matrices), el cual debe ser resuelto para avanzar en el recorrido. Para acceder al panel de control (ActionController), el jugador debe estar en el rango de activación y pulsar la tecla “E”, como se muestra en la Figura 4.29 y Figura 4.30.

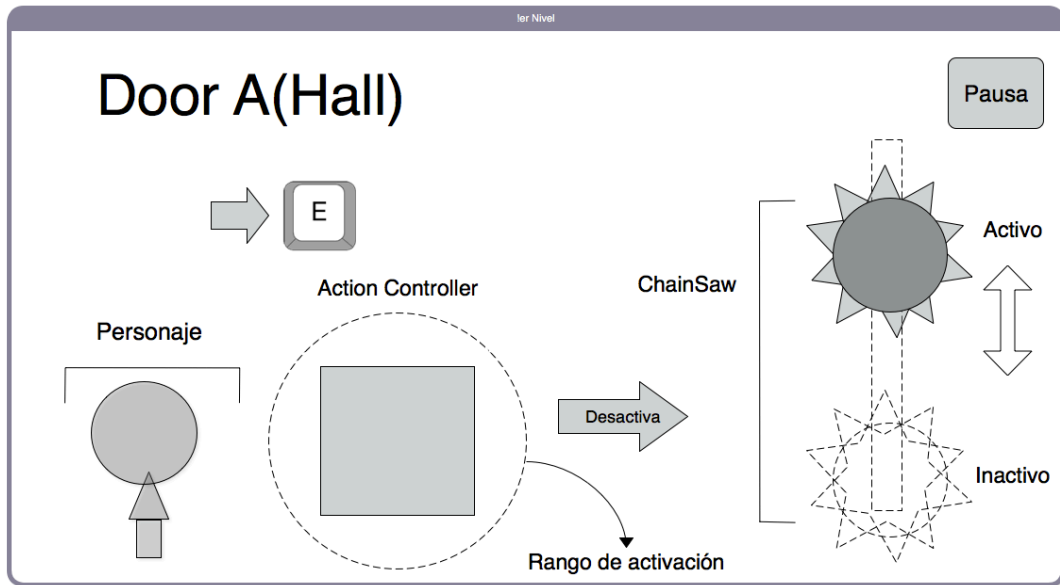


Figura 4.29 Wireframe: Panel de control del primer obstáculo

Fuente: Elaboración propia (2018)

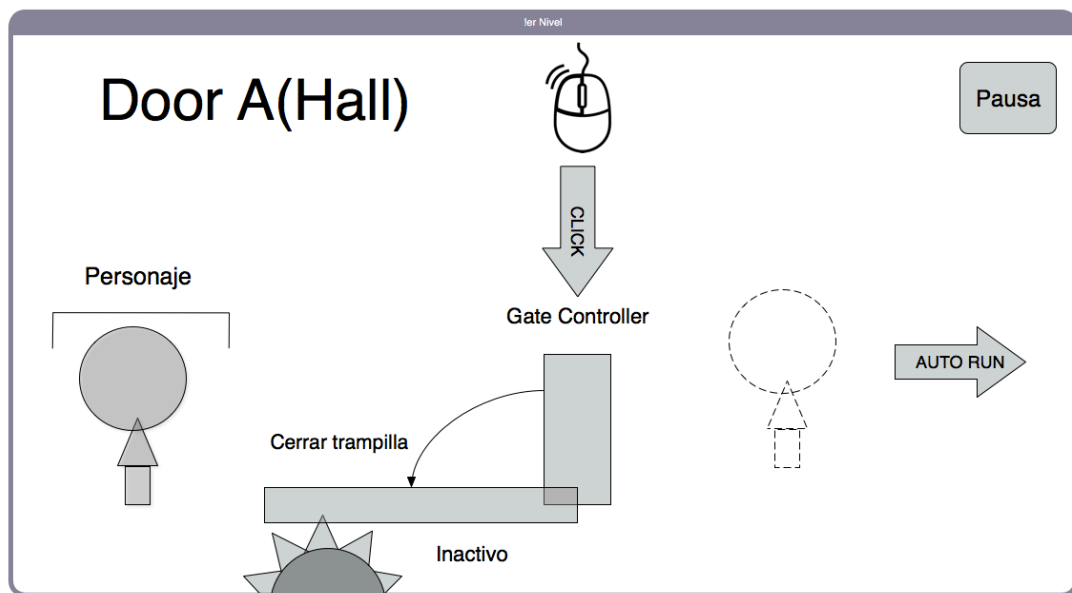


Figura 4.30 Wireframe: Desactivación del primer obstáculo

Fuente: Elaboración propia (2018)

Una vez que el jugador resuelva el puzle asociado al panel de control, este podrá cerrar una trampilla (haciendo clic sobre el GateController) lo que le permitirá avanzar como se puede observar en la imagen 4.30.

Por única vez en el juego, el protagonista estará configurado para correr de forma automática hasta llegar al próximo obstáculo.

f) Puzle de Representación de imágenes

En niveles posteriores, la mecánica (Panel de control → Resolver puzle de matrices → desactivar obstáculo) se mantiene, sin embargo, la complejidad incrementa. En el caso del puzle de representación de imágenes, la matriz gráfica (Panel de imagen) aumenta su orden, pero mantiene la opción de re colorear directamente los píxeles, quiere decir que cuando el jugador hace clic sobre cualquier píxel este puede cambia de color (Blanco a Negro y viceversa).

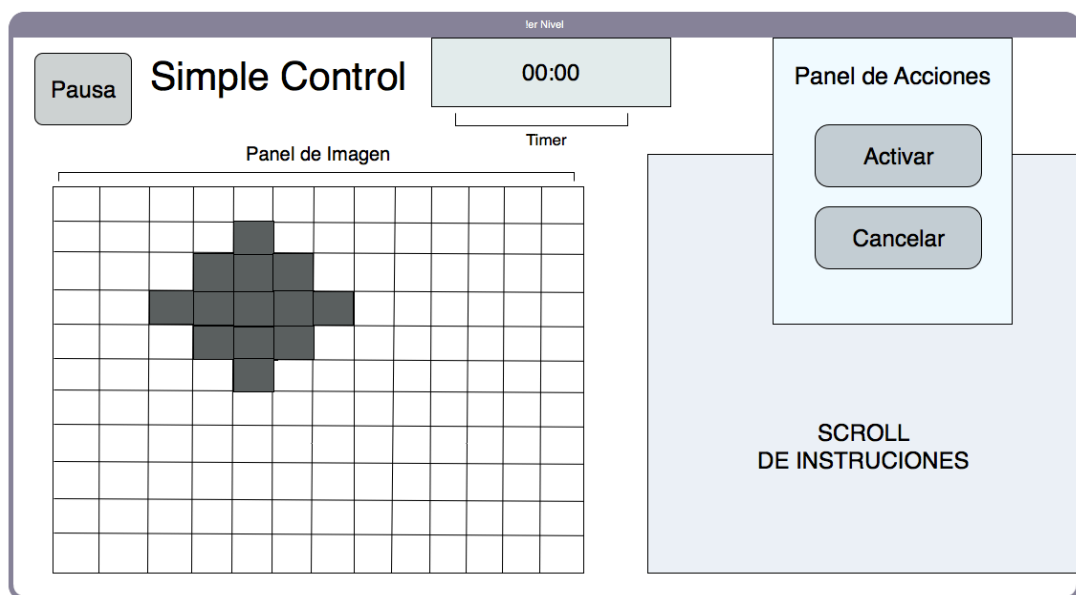


Figura 4.31 Wireframe: Puzle de representación de imágenes (Avanzado)

Fuente: Elaboración propia (2018)

El botón **Activar** para este puzle comprueba que la matriz numérica (que se muestra en el “**Scroll de instrucciones**”) y la gráfica coinciden (Panel de Imagen). De ser así, el jugador podrá continuar en el recorrido.

g) Puzle de Traslación Geométrica

Este puzle integra la lógica para realizar una traslación geométrica en la matriz de la imagen.

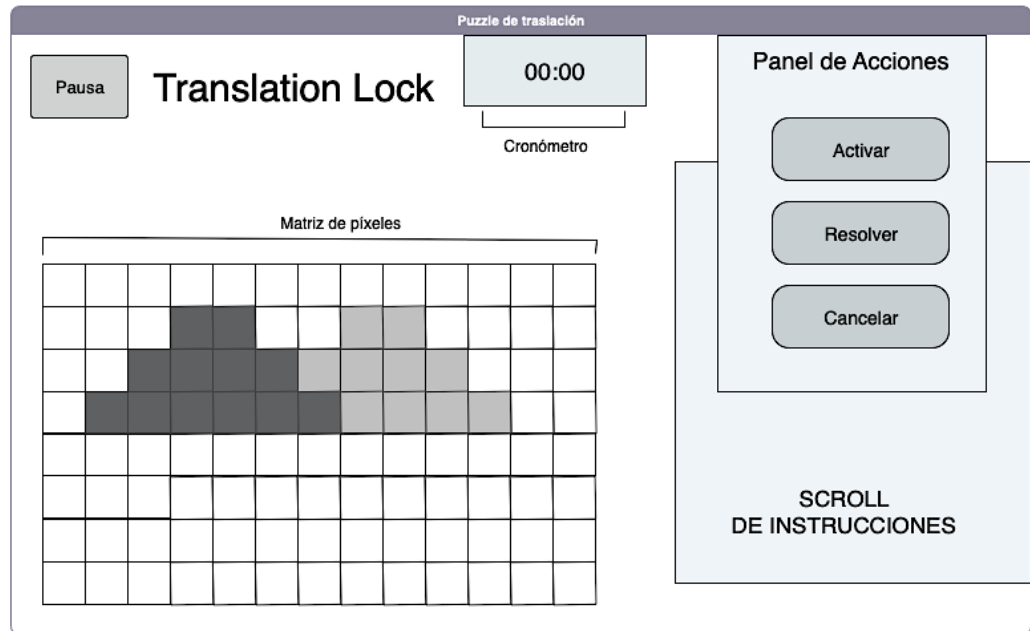


Figura 4.32 Wireframe: Puzle de traslación geométrica

Fuente: Elaboración propia (2018)

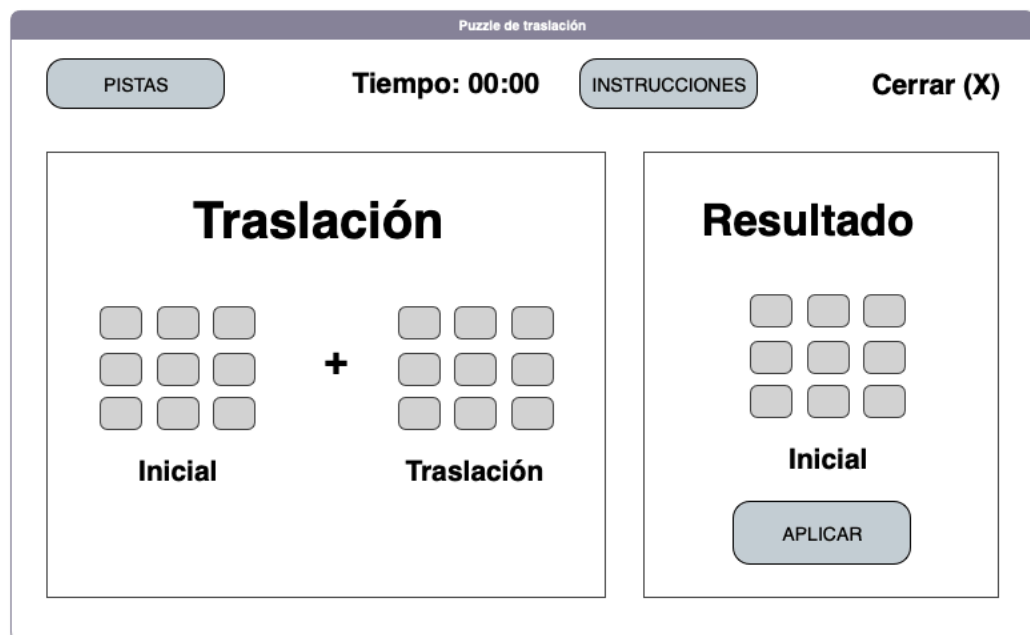


Figura 4.33 Wireframe: Resolución de una traslación geométrica

Fuente: Elaboración propia (2018)

El objetivo básicamente consta en mover la imagen n cantidad de píxeles sobre el eje x o y (dependiendo al ejercicio propuesto). A diferencia del puzzle anterior (Representación de imágenes) este no permite re-colorear la matriz grafica. Entonces para este caso, se realiza una transformación rígida, quiere decir que no deforma la imagen principal.

Como se ve en la Figura 4.33, al aplicar una suma de matrices sobre la matriz numérica inicial esta solo se repositona en el plano.

h) Puzzle de Rotación Geométrica

Este puzzle comprende la lógica para realizar una rotación geométrica en una imagen. El objetivo principal consiste en alterar la posición de un punto en el plano, el cual es rotado alrededor de su origen de coordenadas.

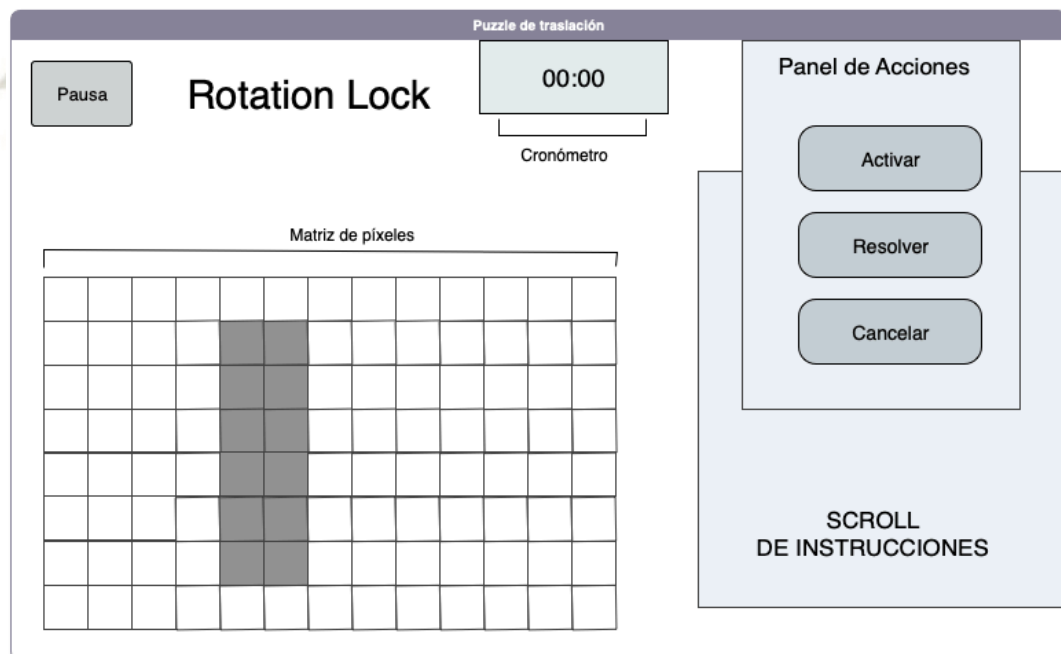


Figura 4.34 Wireframe: Puzzle de rotación Geométrica

Fuente: Elaboración propia (2018)

Para este caso en particular, hacemos el uso de coordenadas homogéneas descritas en la sección “Aspectos disciplinarios”; esto permite tratar una transformación en base a una multiplicación de matrices.

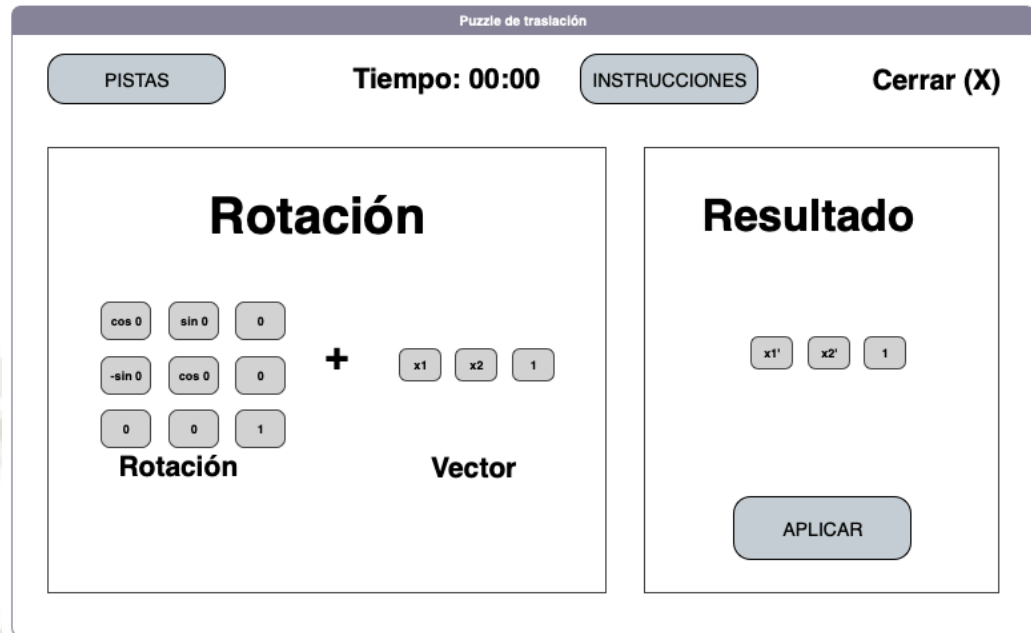


Figura 4.35 Wireframe: Resolución de una rotación Geométrica
Fuente: Elaboración propia (2018)

i) Puzle de Escalamiento Geométrico

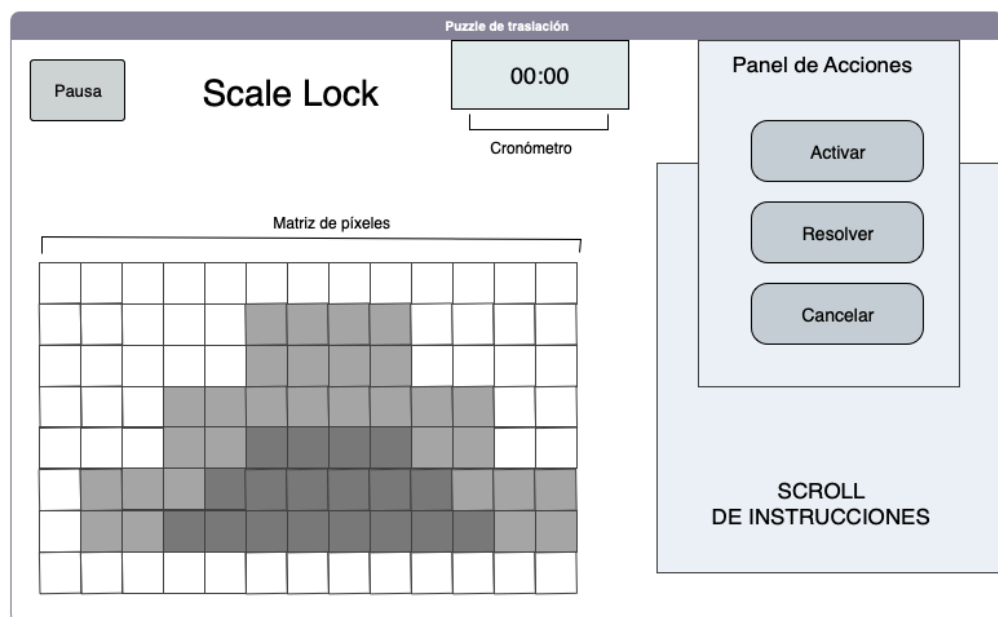


Figura 4.36 Wireframe: Puzle de escalamiento geométrico
Fuente: Elaboración propia (2018)

Este puzzle integra el contenido de escalamiento geométrico, el objetivo del juego es incrementar equitativamente el tamaño de la imagen aplicando el procedimiento de coordenadas geométricas como se detalló en la sección de aspectos didácticos del presente trabajo.

j) **Puzzle de Suma de matrices TDL:** El siguiente puzzle comprende el procedimiento para resolución de algoritmos aplicando el enfoque Test-Driven Learning.

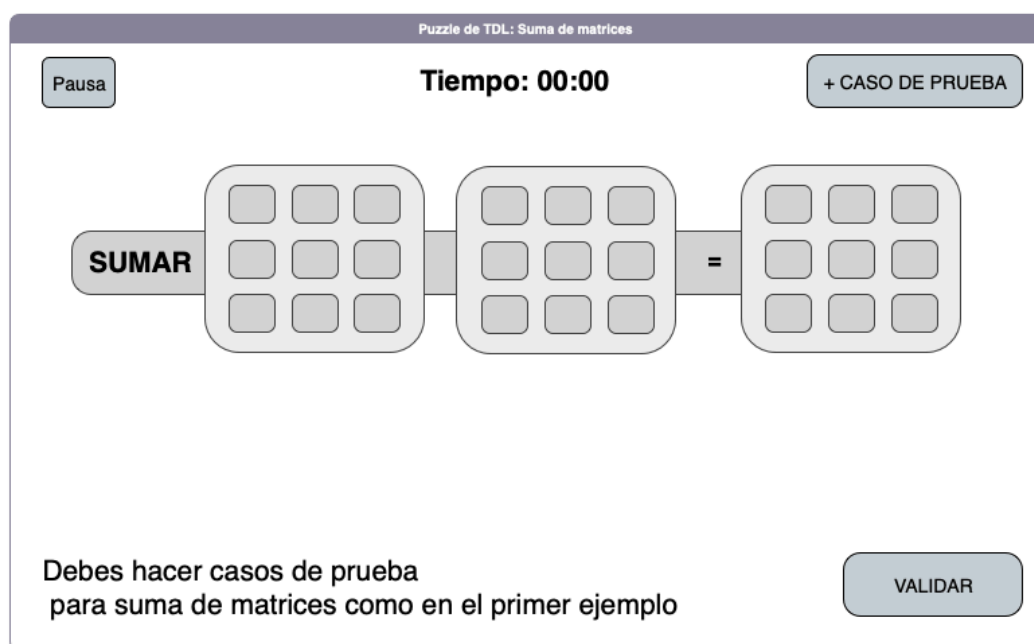


Figura 4.37 Wireframe: Puzzle TDL Casos de prueba para Suma de matrices

Fuente: Elaboración propia (2018)

Inicialmente se le muestra al jugador un entorno de trabajo para realizar casos de prueba, los cuales permiten al jugador establecer una cobertura de pruebas. En este caso, se le pide al usuario diseñar los casos de prueba para una suma de matrices.

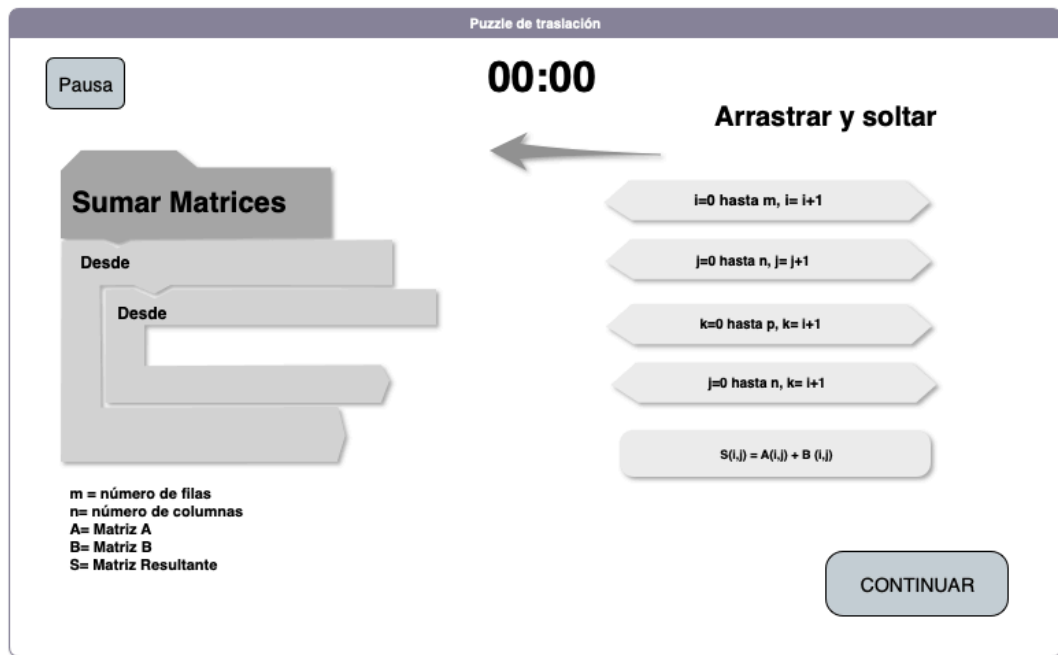


Figura 4.38 Wireframe: Puzzle TDL Implementación Suma de matrices

Fuente: Elaboración propia (2018)

Una vez validados los casos de prueba, el jugador procede a realizar la implementación. Dado que aún los estudiantes no conocen un lenguaje de programación en particular harán uso de la programación con bloques (Scratch).

4. Implementación de software

4.1. Tecnologías Utilizadas

Se optó para la implementación del Juego Serio las siguientes tecnologías de desarrollo:

a) Motor de juego

Unity es un motor de juegos que provee todas las características necesarias para la construcción de videojuegos (Manejador de escenarios, Creador de instaladores, Animaciones, IDE programación, etc). Dado que esta herramienta mantiene una curva de aprendizaje media, está dentro de los softwares más populares para este fin. Acerca de sus funcionalidades.

Unity permite renderizar gráficos 3D y 2D (para el presente proyecto se optó por el entorno 2D). Así mismo, también provee de un motor de física, el cual simula todas las leyes de la física requeridas (gravedad, fricción, movimiento, etc) para la creación de videojuegos. Unity también permite realizar animaciones directamente desde su entorno de desarrollo, esto facilita la centralización de los elementos del juego en el proyecto.

b) Lenguaje de programación

El motor de juegos Unity soporta los lenguajes: C# y Javascript. Se optó seleccionar el lenguaje de programación C# dado que la mayoría de las librerías que se usaron en el proyecto estaban construidas en dicho lenguaje.

c) Diseño gráfico

Para la creación del contenido gráfico (Assets) se utilizó el programa **Affinity Designer** (uso de licencia personal). Este software permite el diseño gráfico en vectores, se utilizó para la creación de los Sprites Sheets del proyecto (Imágenes sueltas que componen un objeto de juego con la finalidad de su posterior animación) y los prototipos finales.

d) Backend (Web Services)

Sada su facilidad al desplegarse en un servidor Linux, para la construcción de los web-services del proyecto se utilizó el framework Laravel en su versión 7.3 en base al lenguaje de programación PHP 7.

e) Base de datos

Para el presente proyecto se utilizó una base de datos en MySQL 5.3. Para el acceso a esta, se optó por la interfaz de código abierto: Sequel Pro, el cual permite realizar consultas de manera remota en la base de datos.

4.2. Configuración inicial

Antes de iniciar la etapa de implementación del juego se realizó la configuración del entorno de desarrollo y de los ambientes para la creación del contenido. A continuación, se describe el proceso de cada uno de ellos.

4.2.1. Instalación y configuración del motor de juegos

Para descargar el motor de videojuegos se debe ir a la página oficial de descarga: store.unity.com/download, en la parte inferior aceptamos los términos y condiciones, y se procede con la descarga de Unity Hub (Repositorio de Unity para múltiples versiones). Ejecutamos el archivo como se observa en la Figura 4.39.

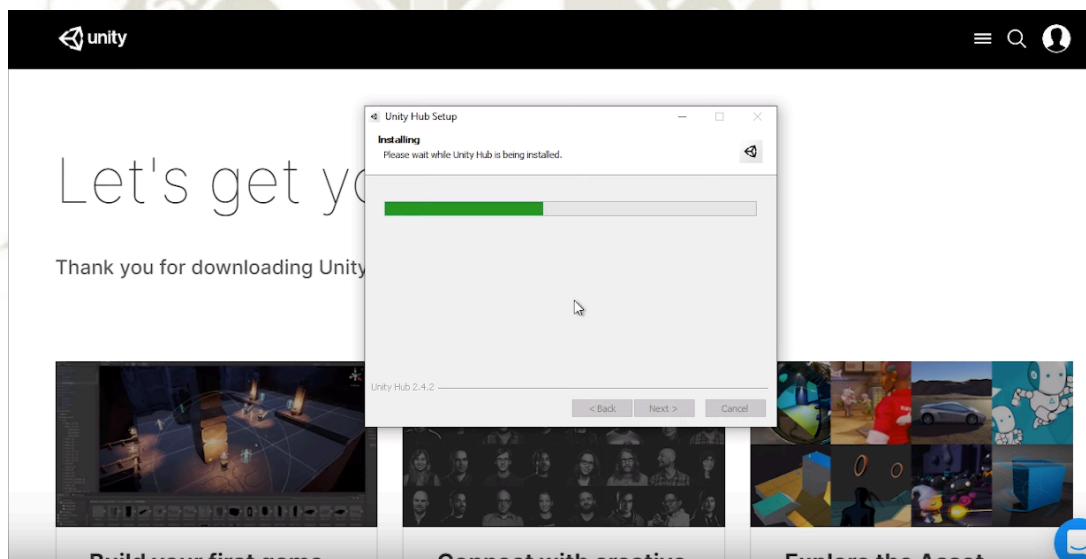


Figura 4.39 Instalación de Unity Hub

Fuente: Elaboración propia (2018)

Una vez finalizado el proceso de instalación se abrirá el panel de proyectos recientes, sin embargo, para crear un nuevo proyecto se requiere de una cuenta para unity, la cual puede ser obtenida de forma gratuita desde el mismo programa.

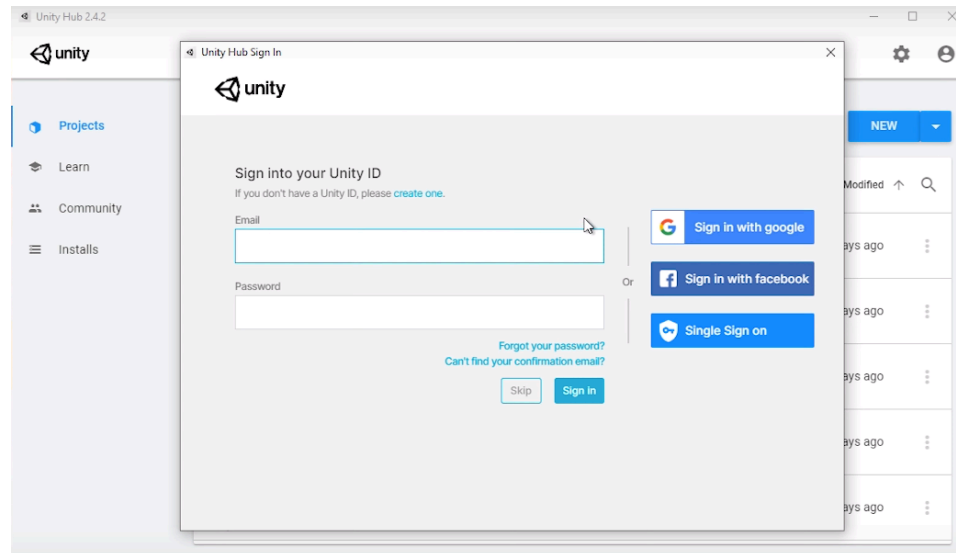


Figura 4.40 Configuración inicial Unity Hub

Fuente: Elaboración propia (2018)

Posterior a la creación de la cuenta, se procede con la instalación de la versión con la que se construirá el entorno de trabajo. Para ello en la misma ventana, en el menú seleccionamos la opción “Installs” como se muestra en la Figura 4.41.

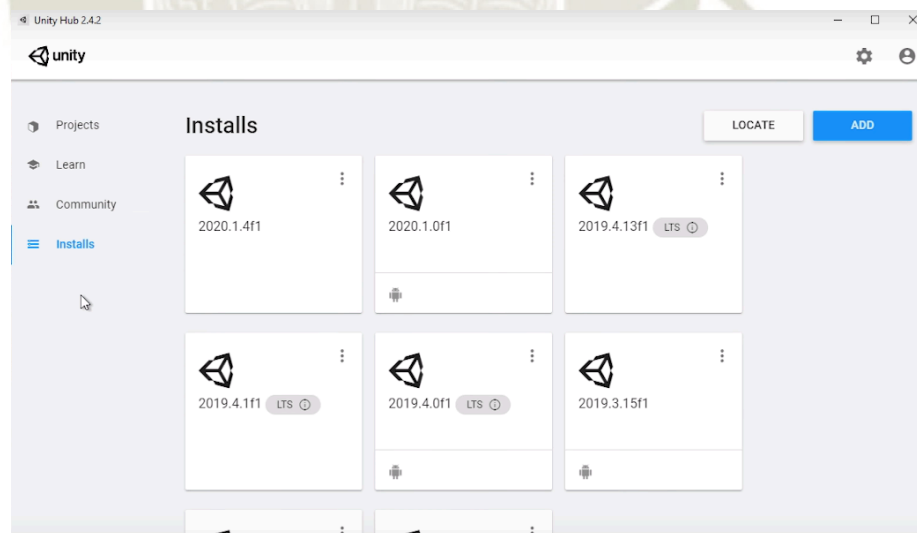


Figura 4.41 Listado de versiones disponibles de Unity

Fuente: Elaboración propia (2018)

En esta sección se mostrarán todas las versiones instaladas en el computador. Para añadir una nueva versión pulsamos el botón de añadir y

seleccionamos la versión más reciente (es recomendable utilizar una versión con el terminal “f” lo cual indica que es una versión final).

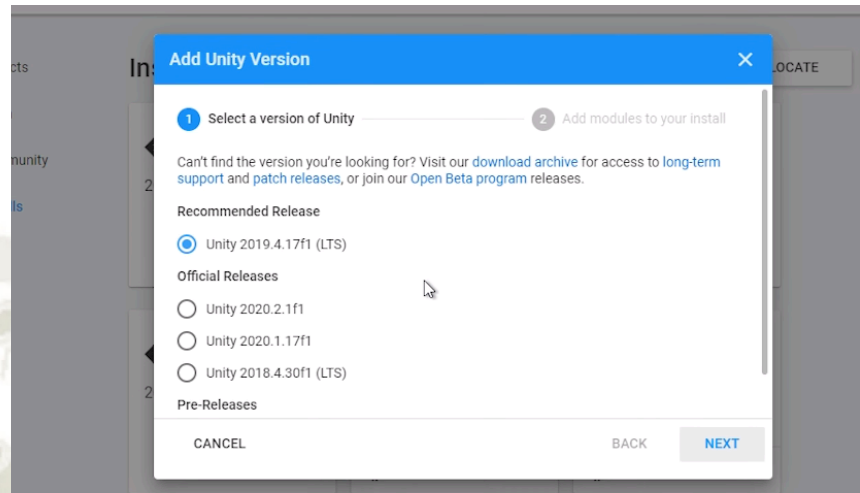


Figura 4.42 Listado de versiones disponibles de Unity

Fuente: Elaboración propia (2018)

Después de la selección de la versión, es necesario añadir ciertos módulos que permiten la distribución del juego para distintas plataformas (IOS, Android, Web Gl, Windows/Mac), para el proyecto se seleccionó (Web Gl y Windows), además del IDE de programación, que para este caso fue “Visual Studio Code”. Finalizada la descarga, ya es posible la creación de nuevos proyectos, para ello el panel de creación mostrará un listado de entornos como se muestra a continuación.

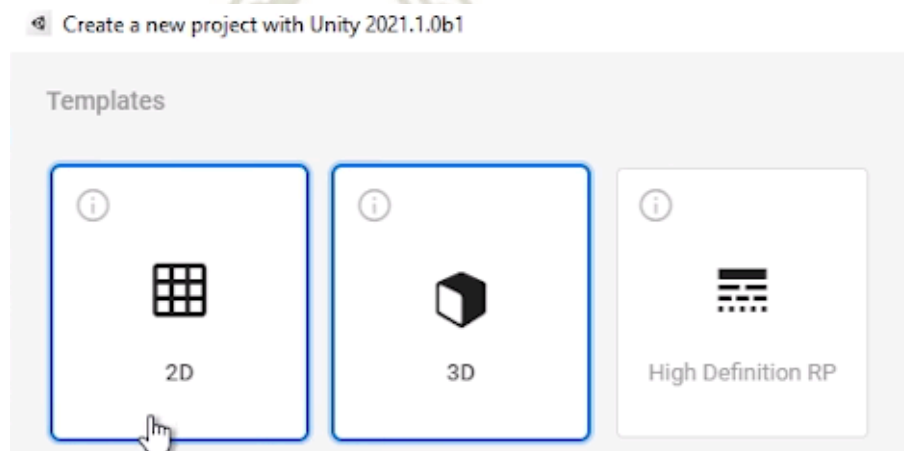


Figura 4.43 Creación del proyecto para el Juego Serio

Fuente: Elaboración propia (2018)

Para el caso de este proyecto se seleccionó en entorno 2D, bajo el nombre de “Engineers Escaping” y se procedió con la carga para iniciar la construcción del Juego Serio. El setup final es como se muestra en la Figura 4.44 (Se utilizó el layout de la interfaz por defecto).

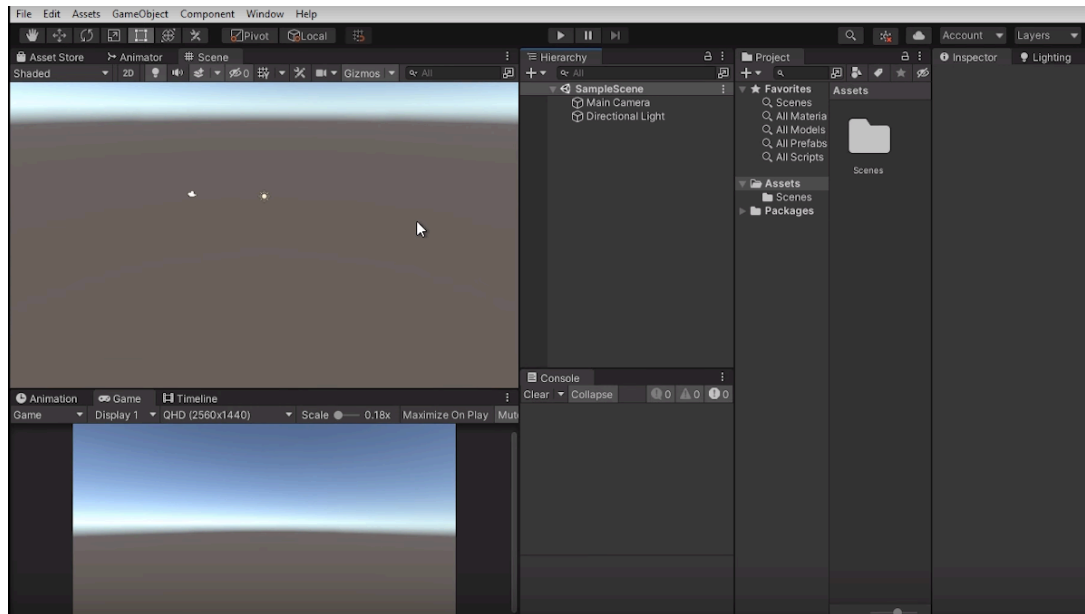


Figura 4.44 Entorno inicial del motor de juegos Unity
Fuente: Elaboración propia (2018)

4.2.2. Instalación y configuración del entorno de desarrollo (Backend)

La instalación del servidor y base de datos en un entorno de desarrollo se puede realizar a través de MAMP (en el caso de sistemas operativos de Apple) y WAMP (en el caso de sistema operativo Windows). En ambos casos se establece un servidor local con Apache, MySql y PHP. Para este proyecto se utilizó MAMP PRO (Licencia de uso personal) el cual se obtuvo desde el repositorio oficial este se configuró con los puertos 80\443 SSL en Apache y 3306 en MySql.

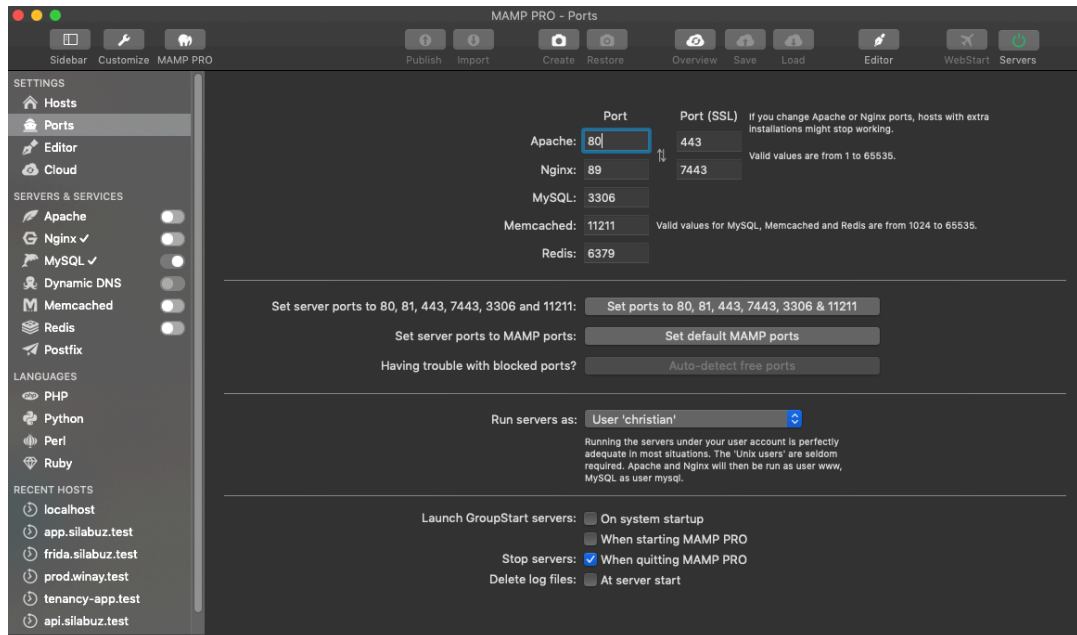


Figura 4.45 Configuración del servidor local

Fuente: Elaboración propia (2018)

A continuación, se describe el proceso de la creación del proyecto en Laravel 7 para el desarrollo del backend.

a) Proceso de instalación del manejador de paquetes de PHP “composer”

- Inicialmente se inició con la descarga el paquete con el siguiente comando:

```
$ php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
```

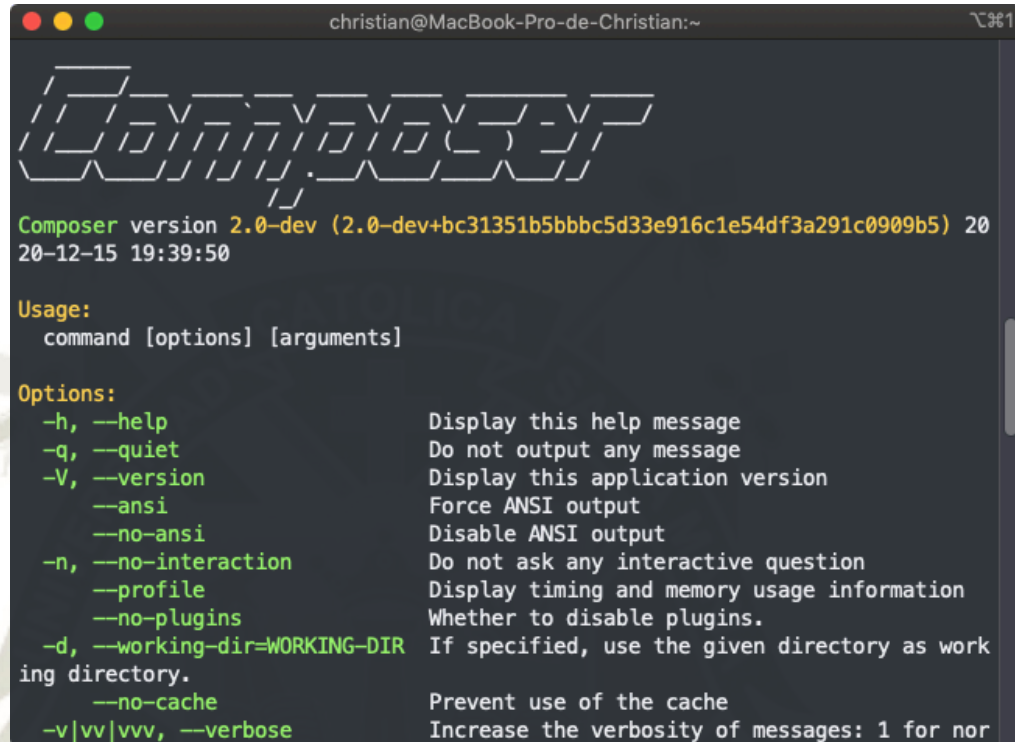
- Posterior a la descarga se procedió con la instalación del paquete de manera global

```
$ php composer-setup.php --install-dir=/usr/local/bin --filename=composer
```

- Una vez concluido, se removió el instalador a la papelerera de reciclaje

```
$ php -r "unlink('composer-setup.php');"
```

- Para verificar que la instalación se hizo de manera correcta, se ingresa el comando: “composer” desde cualquier directorio de la consola. El resultado debe ser el siguiente:



```
christian@MacBook-Pro-de-Christian:~  
  
Composer version 2.0-dev (2.0-dev+bc31351b5bbbc5d33e916c1e54df3a291c0909b5) 20  
20-12-15 19:39:50  
  
Usage:  
  command [options] [arguments]  
  
Options:  
  -h, --help                Display this help message  
  -q, --quiet               Do not output any message  
  -V, --version             Display this application version  
      --ansi                Force ANSI output  
      --no-ansi             Disable ANSI output  
  -n, --no-interaction     Do not ask any interactive question  
      --profile             Display timing and memory usage information  
      --no-plugins          Whether to disable plugins.  
  -d, --working-dir=WORKING-DIR If specified, use the given directory as work  
ing directory.  
      --no-cache            Prevent use of the cache  
  -v|vv|vvv, --verbose     Increase the verbosity of messages: 1 for nor
```

Figura 4.46 Instalación del Manejador de paquetes PHP :Composer

Fuente: Elaboración propia (2018)

b) Instalación y configuración del proyecto en Laravel

Una vez instalado **composer**, se procede a instalar el comando **laravel** de manera global, este comando permite crear un nuevo proyecto de laravel con todas las dependencias necesarias, en una estructura MVC.

Para realizar este procedimiento ingresamos los siguientes comandos.

```
$ composer global require laravel/installer
```

```
$ laravel new api-engineers-escaping
```

```

laravel new api-engineers-escaping
┌───────────┐
│ Laravel   │
└───────────┘

Warning: This development build of composer is over 60 days old. It is recommended to update it by running "/usr/local/bin/composer self-update" to get the latest version.
Creating a "laravel/laravel" project at "./api-engineers-escaping"
Installing laravel/laravel (v8.5.17)
- Downloading laravel/laravel (v8.5.17)
- Installing laravel/laravel (v8.5.17): Extracting archive
Created project in /Users/christian/Projects/api-engineers-escaping
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 104 installs, 0 updates, 0 removals
- Locking asm89/stack-cors (v2.0.3)
- Locking brick/math (0.9.2)
- Locking doctrine/inflector (2.0.3)
- Locking doctrine/instantiator (1.4.0)
- Locking doctrine/lexer (1.2.1)

```

Figura 4.47 Instalación de. Manejador de paquetes PHP:Composer

Fuente: Elaboración propia (2018)

c) Creación del virtual host para el proyecto en Laravel

Finalmente, configuramos el virtual host en MAMP bajo el nombre “api.engineers-escaping.test” con la finalidad de acceder directamente al dominio temporal desde el navegador.

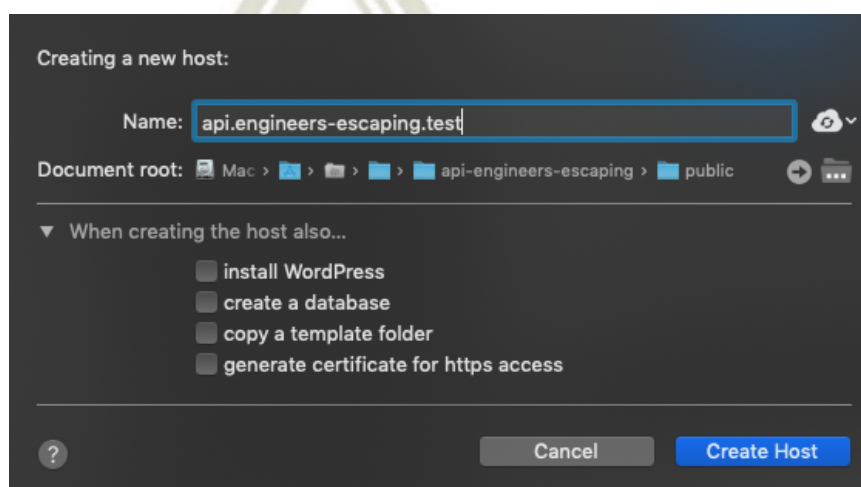


Figura 4.48 Creación del hostname para un entorno desarrollo

Fuente: Elaboración propia (2018)

4.3. Creación del contenido audiovisual

Para el proyecto en cuestión y de acuerdo con el diseño de interfaces y niveles, en este punto se describe el proceso de creación de assets, los cuales parten de una hoja de cortes (Spritesheet).

a) Personaje principal

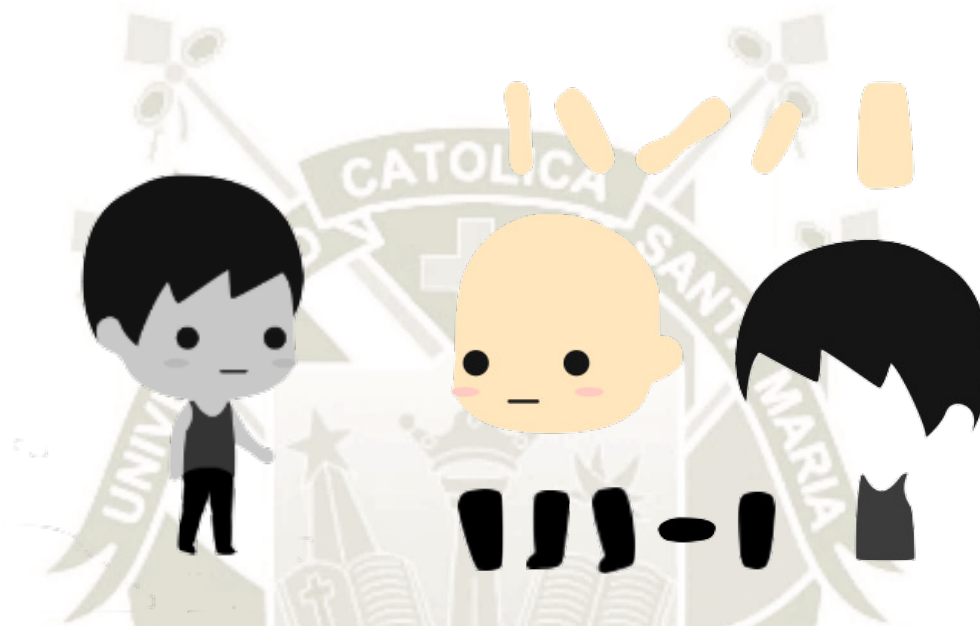
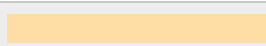

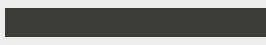


Figura 4.49 Spritesheet: Personaje Principal

Fuente: Elaboración propia (2018)

Tabla 4.35

Paleta de colores – Personaje Principal

Sección	Código Hexadecimal	Color
Piel	#FFDEA5	
Cabello	#101010	
Polo	#3E3C39	

Fuente: Elaboración propia (2018)

b) Trampas y obstáculos del juego

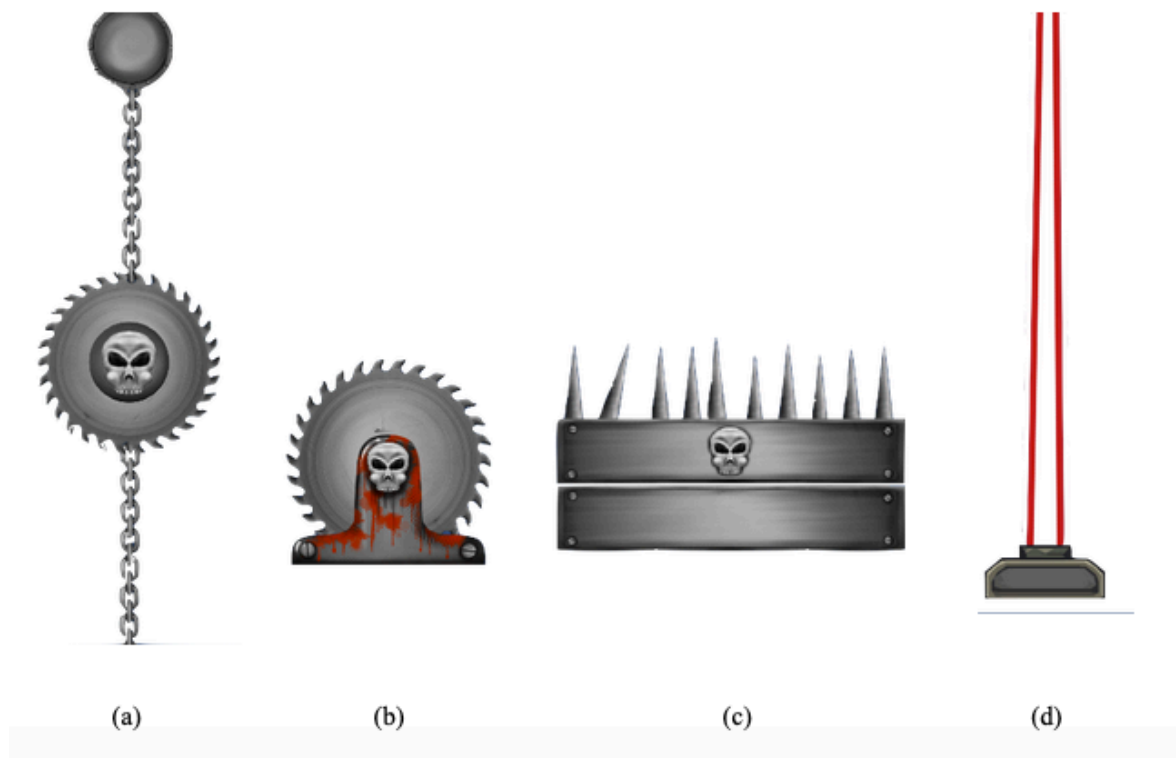


Figura 4.50 Spritesheet: Obstáculos del juego

Fuente: Elaboración propia (2018)

c) Personajes secundarios



Figura 4.51 Spritesheet: Personaje secundario

Fuente: Human Sprites Package

d) Componentes de los puzles

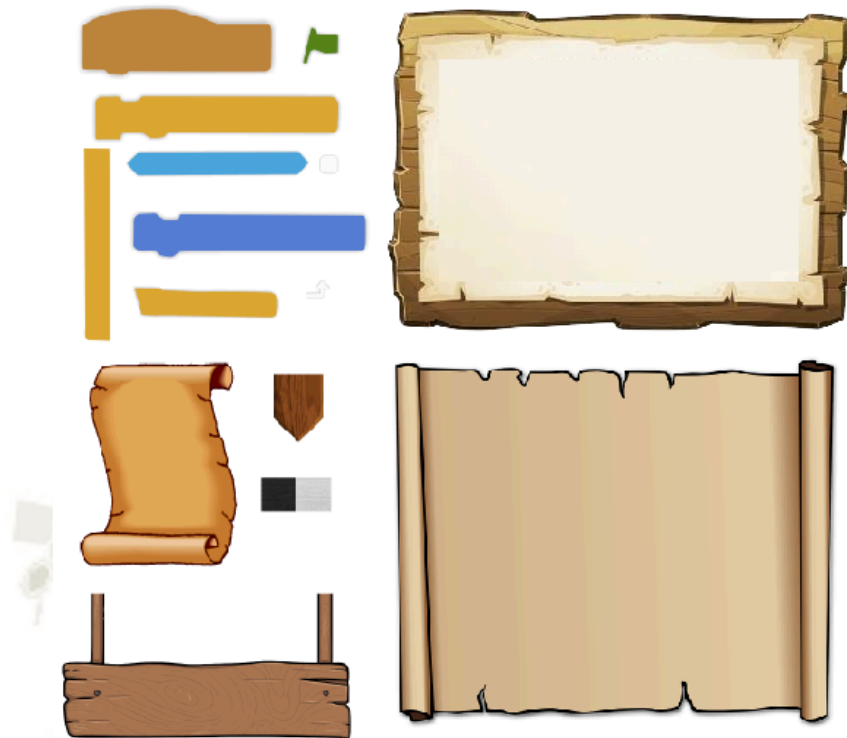


Figura 4.52 Spritesheet: Componentes para los puzles

Fuente: Human Sprites Package

e) Fondo principal



Figura 4.53 Imagen para el fondo principal

Fuente: Sci-fi 2D Asset's package

f) Iconografía



Figura 4.54 Iconografía del juego

Fuente: Elaboración propia (2018)

4.4. Desarrollo del Backend

En esta sección se detalla los aspectos técnicos de la construcción de la capa de lógica de negocio y acceso a datos, comprende todos los procesos y acciones que permiten el intercambio de información entre la base de datos y el Juego Serio.

4.4.1. Modelos y migraciones

Al inicio del desarrollo del proyecto es importante definir las migraciones, este mecanismo permite llevar un control de versiones de la base de datos de acuerdo con la arquitectura planteada.

En Laravel, para crear un archivo de migración se debe ejecutar el siguiente comando desde la terminal del proyecto:

```
$ php artisan make:migration create_%table_name%_table
```


A continuación, se muestra un extracto de las migraciones más relevantes del proyecto.

```
Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('code', 10);
    $table->enum('gender', ['masculino', 'femenino']);
    $table->enum('status', ['active', 'disabled']);
    $table->string('email')->unique();
    $table->string('group');
    $table->timestamp('email_verified_at')->nullable();
    $table->string('password');
    $table->rememberToken();
    $table->timestamps();
});
```

Figura 4.55 Migración de la entidad de Usuarios (Users)

Fuente: Elaboración propia (2018)

```
Schema::create('chapters', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->float('max_score');
    $table->timestamps();
});
```

Figura 4.56 Migración de la entidad de Capítulos (Chapters)

Fuente: Elaboración propia (2018)

```
Schema::create('attempts', function (Blueprint $table) {
    $table->id();
    $table->bigInteger("user_id");
    $table->bigInteger("chapter_id");
    $table->float("score");
    $table->integer("deaths");
    $table->duration("time");
    $table->timestamps();
    // users_attempts_relationship
    $table->foreign('user_id')->references('id')->on('users');
    $table->foreign('chapter_id')->references('id')->on('chapters');
});
```

Figura 4.57 Migración de la entidad de Intentos (Attempts)

Fuente: Elaboración propia (2018)

```
Schema::create('settings', function (Blueprint $table) {  
    $table->id();  
    $table->text('allowed_ips');  
    $table->enum('server_status', ['active', 'stopped']);  
    $table->timestamps();  
});
```

Figura 4.58 Migración de la entidad de Configuraciones (Settings)

Fuente: Elaboración propia (2018)

Como se puede observar en las figuras anteriores (Figura 4.55, Figura 4.56, Figura 4.57 y Figura 4.58), las migraciones también permiten crear relaciones entre tablas, lo cual facilita la creación del esquema de base de datos. Una vez creado el esquema de base de datos, se procedió a crear las clases de modelos con la ayuda del comando:

```
$ php artisan make:model %Model_name%
```

Un modelo representa una abstracción para una tabla de base de datos determinada, por lo general se crea el directorio App/Models del proyecto. Los modelos se integran con Eloquent (un generador de consultas), el cual permite realizar acciones directamente a los datos desde la capa de lógica de negocio.

4.4.2. Controladores

A continuación, se presentan los controladores, aquellos componentes de software que reciben y atienden las peticiones del cliente. Los controladores son los encargados de utilizar la lógica de los modelos para realizar una determinada acción. Estos usualmente se almacenan en la carpeta App/Http/Controllers. Para crear un controlador se utiliza el siguiente comando:

```
$ php artisan make:controller %NameController%
```

Un controlador de tipo Resource es aquel que integra las acciones básicas (CRUD) para el tratamiento de datos sobre un modelo en particular. Para crear un controlador de este tipo en Laravel, añadimos la directiva: -resource al comando anterior. Cuando un controlador es muy complejo, o solo requiere de una acción en específico, se utilizan los Single Action Controllers. Para ello solo se debe definir una única función llamada `__invoke()` dentro del controlador.

A continuación, se describen cada uno de los controladores utilizados en el proyecto:

Tabla 4.36

Lista de controladores: Backend

Nombre	Tipo	Descripción
UserController	Resource	Comprende todas las acciones que modifican el modelo de usuarios (Crear, Editar, Leer y Eliminar)
APIController	Basic	Agrupar las peticiones con acciones específicas del cliente (Ver ranking, y Ver capítulos disponibles)
AttemptController	Single	Solo realiza la acción de almacenar un registro en la tabla de intentos (Attempts).
SettingsController	Basic	Comprende las configuraciones globales del juego (Habilitar/Deshabilitar el juego y Permitir conexiones del cliente)

Elaboración propia (2018)

4.4.3. Seguridad y acceso a datos

Para garantizar seguridad y el control de acceso del juego solo a los usuarios autorizados, Laravel provee de un paquete de librerías denominado como Sanctum que permite gestionar la autenticación y la protección de rutas en sistemas SPA (Single web applications) basadas en cookies de sesión y Web Services (API) basadas en tokens. El paquete de librerías Laravel\Sanctum permite a cada usuario de la aplicación pueda generar sus tokens para su cuenta de usuario.

Para el presente proyecto se optó por utilizar el sistema de autenticación por tokens (Personal Access Tokens). Dado que el cliente (Juego Serio) no solo fue desarrollado para plataformas Web, sino que también para plataformas de escritorio y esta no permite el almacenamiento de cookies.

A continuación, se describe el proceso recabado:

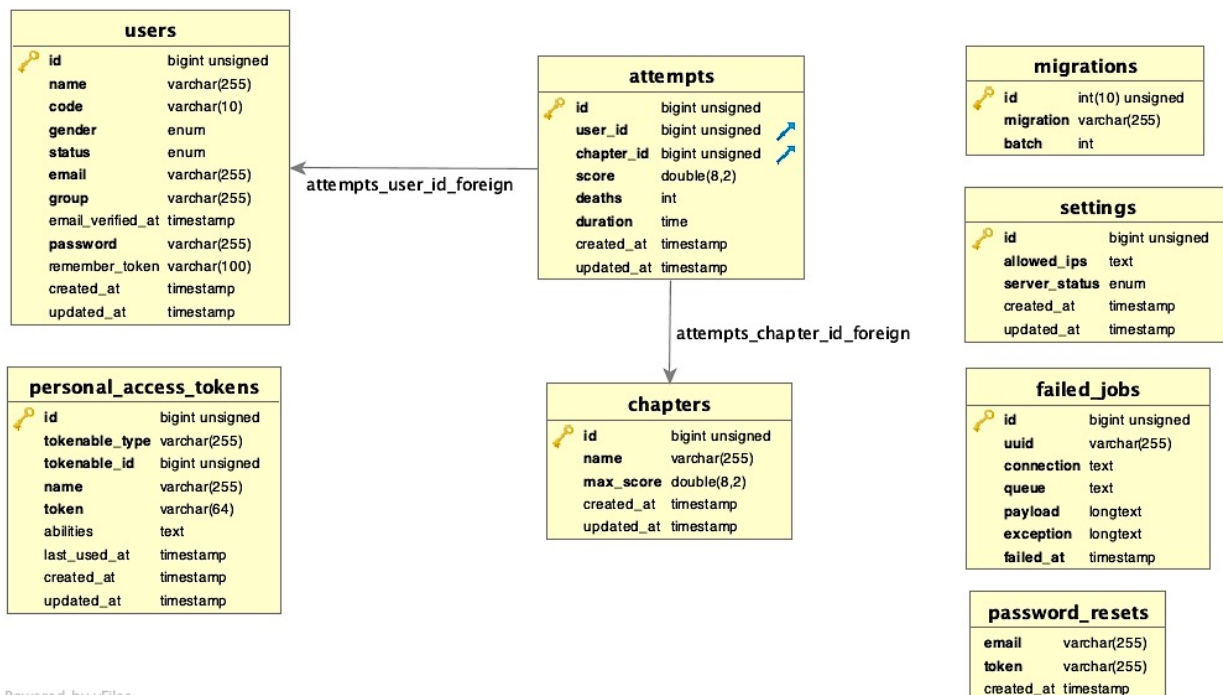
a) **Instalación**

La instalación de Sanctum se realiza utilizando el manejador de paquetes de PHP previamente instalado en la etapa de configuración inicial. Para realizar la instalación, ejecutamos el siguiente comando desde la terminal del proyecto:

```
$ composer require laravel/sanctum
```

Esta acción descargará los recursos y dependencias que utiliza Sanctum y los copiará en la carpeta **vendor** (Librerías del proyecto).

Esta acción crea las tablas donde se almacenan los Tokens de los usuarios.



Powered by yFiles

Figura 4.59 Esquema final de base de datos (Backend)

Fuente: Elaboración propia (2017)

b) Configuración

Para comenzar a generar Tokens para los usuarios, se debe editar el modelo `App\Models\User` y añadir el trait: `Laravel\Sanctum\HasApiTokens`.

```
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasFactory, Notifiable, HasApiTokens;
}
```

Figura 4.60 Habilitación del Usuario para la generación de API Tokens

Fuente: Elaboración propia (2018)

Esto permite al modelo `User` utilizar los métodos para gestionar los tokens de usuario. Dado que no se ha planteado en el diseño el acceso por Roles, se omite el uso de Scopes (Delimitación de recursos basado en etiquetas). A continuación, se detalla el método de autenticación utilizado basado en tokens del proyecto.

```
public function login (Request $request)
{
    $credentials = $request->only('code', 'password');
    $user = User::getUserByEmail($credentials['email']);

    if (Hash::check($request->password, $user->password)) {
        // Authentication passed...

        $token = $user->createToken('app-game-access')->plainTextToken;
        $response = [
            'user' => $user->getData(),
            'token' => $token
        ];
        return response($response,201);
    }
    else
    {
        return response()->json(['message'=>'Unauthorized'],403);
    }
}
```

Figura 4.61 Método para la autenticación basado en tokens personales

Fuente: Elaboración propia (2018)

c) Protección de rutas

Para ser efectiva la protección de rutas se debe añadir el middleware: `auth:Sanctum` en los controladores que requieran un control de acceso. Esto se puede realizar directamente en la hoja de rutas (`routes/api.php`), sin embargo, para este proyecto se optó por definirlo por convención en el constructor de cada controlador, como se puede ver a continuación.

```
class UserController extends Controller
{
    public function __construct(){
        $this->middleware('auth:sanctum');
    }
}
```

Figura 4.62 Protección de rutas desde del controlador de usuarios

Fuente: Elaboración propia (2018)

En el caso de los recursos que no requieren autenticación como: `Login()`, `Register()`, `ServerStatus()`, donde el usuario se encuentra en una etapa previa al inicio de sesión, se añaden como excepciones en el constructor del controlador. Esto se puede ver en la Figura 4.63:

```
class APIController extends Controller
{
    public function __construct(){
        $this->middleware('auth:sanctum')
        ->except(['login', 'register', 'server_status']);
    }
}
```

Figura 4.63 Protección de rutas con excepciones desde el controlador de recursos

Fuente: Elaboración propia (2018)

4.4.4. Web Services

A continuación, se listan todas las rutas (API web services) junto a sus métodos de accesos disponibles para el consumo del cliente (Juego Serio: Engineers Escaping), así como también el controlador al que pertenecen.

Tabla 4.37

Lista de rutas API: Backend

Tipo	Ruta	Nombre	Controlador
GET	/api/users	users.index	UserController@index()
GET	/api/users/{id}	users.show	UserController@show(\$id)
POST	/api/users	users.store	UserController@store(\$request)
PUT	/api/users	users.update	UserController@update(\$request)
GET	/api/auth	users.auth	UserController@auth()
POST	/api/login	api.login	ApiController@login(\$request)
POST	/api/register	api.register	ApiController@register(\$request)
POST	/api/attempts	attempts.store	AttemptController@__invoke()
GET	/api/server-status	Settings.server	SettingController@serverStatus()

Elaboración propia (2018)

4.5. Desarrollo de componentes de juego

En esta sección se describen los componentes (Scripts, Clases y Controladores) que forman parte de la construcción del Juego Serio.

4.5.1. Motor de cálculo (Matrix Engine)

Este componente es el encargado de realizar los cálculos necesarios para la validación y la evaluación de los puzles en los juegos de lógica. Las clases se desarrollaron en base al modelo lógico expuesto en la etapa de diseño de software.

A continuación, se describe las interfaces de cada una de ellas:

Clase Punto (Point)

Esta clase define un punto en plano, comprende un objeto del tipo `Vector2` de la librería de Unity. Esta clase se define bajo dos constructores (por defecto y con valores iniciales) como se puede apreciar en la Figura 4.64.

```
public class Point (){\n    private Vector2 point;\n    public Point(){\n        this.point = new Vector2();\n    }\n    public Point(int x , int y){\n        this.point = new Vector2(x,y);\n    }\n    public cross(Point p){\n        return point.x * p.y - point.y * p.x;\n    }\n}
```

Figura 4.64 Interfaz de la clase Coordenada (Point)

Fuente: Elaboración propia (2018)

Dado que la clase `Vector2` no define un método que permitiese realizar un producto cruzado entre dos vectores. Esta clase lo implementa a través del método `cross()` el cual da como resultado la magnitud del vector que perpendicular del producto con otro punto.

Clase Píxel (Box)

Esta clase representa a un píxel de una imagen. Esta clase comprende a) Un objeto del tipo `GameObject`: el cual hace referencia al recurso grafico (Sprite) del píxel (cuadro blanco), b) Un Objeto del tipo **Color**: El cual define el color del `gameObject`, a través del `SpriteRenderer` de dicho objeto, c) Un atributo numérico (`value`): El cual representa el valor binario del píxel [0 → Blanco, 1 → Negro] y d)

Un objeto del tipo Point: El cual almacena las coordenadas del centro del píxel. En la Figura 4.65 se puede observar la interfaz para esta clase.

```
public class Box {  
    // Attributes  
    private GameObject gameObject;  
    private Color color ;  
    private int value;  
    private Point center;  
    // Methods  
    public Box(){...  
    }  
    public Color COLOR{...  
    }  
    public int VALUE{...  
    }  
    public Point CENTER{...  
    }  
    public GameObject GAME_OBJECT{...  
    }  
}
```

Figura 4.65 Interfaz de la clase Pixel (Box)

Fuente: Elaboración propia (2018)

Clase Matriz (Matrix)

Esta clase genérica representa una matriz de elementos ($m \times n$), contiene un vector bidimensional de elementos tipo T, esto quiere decir que no se especifica para un tipo de dato particular. Así mismo, se definen como métodos estáticos o de clases las operaciones matriciales básicas.

```
public class Matrix<T> {  
    //Attributes  
    public T [,] values;  
    private int rows;  
    private int columns;  
    // Methods  
    public Matrix (int m, int n, bool identity=false){...  
    }  
    //Matrix basic operations  
    public static Matrix Sum( Matrix A, Matrix B){...  
    }  
    public static Matrix Mult(Matrix A, Matrix B){...  
    }  
    // Getters and Setters  
    public int getValue(int x,int y){...  
    }  
    public int setValue(int x,int y, int value){...  
    }  
}
```

Figura 4.66 Interfaz de la clase Matriz (Matrix)

Fuente: Elaboración propia (2018)

Como se puede observar en la figura anterior, la interfaz de esta clase comprende la estructura básica de una matriz genérica, la cual soporta la instanciación como matriz de identidad. Así también, se define como métodos de clase las operaciones para la Suma y Multiplicación de matrices. A continuación, se describe cada una de ellas.

a) Suma de matrices (Sum)

Este método estático permite realizar una suma entre dos matrices que tienen las mismas dimensiones y tipología. La implementación se realizó de manera iterativa, para lo que se utilizó 2 bucles for anidados, como se puede observar en la figura inferior.

```
public static Matrix Sum( Matrix A, Matrix B){
    if(!(A.HEIGHT == B.HEIGHT && A.WIDTH == B.WIDTH)){
        throw new Exception("Matrices doesn't match");
    }
    Matrix S = new Matrix(A.HEIGHT, B.WIDTH)
    for (int i = 0; i < A.HEIGHT; i++) {
        for (int j = 0; j < B.WIDTH; j++) {
            S.setValue(i, j, A.getValue(i, j) + B.getValue(i, j));
        }
    }
    return S;
}
```

Figura 4.67 Implementación del método Suma de matrices

Fuente: Elaboración propia (2018)

b) Multiplicación de matrices (Mult)

Este método estático o de clase permite realizar una multiplicación entre dos matrices, siempre y cuando tengan la misma tipología y se cumpla la regla para el producto de matrices (La cantidad de filas de la primera matriz debe ser igual al número de columnas de B). La implementación de este método es más compleja debido a que la suma de matrices se realizó utilizando 3 bucles for anidados como se puede observar en la Figura 4.68 que se muestra a continuación.

```
public static Matrix Mult(Matrix A, Matrix B){
    if(!A.HEIGHT == B.WIDTH){
        throw new Exception("Matrices doesn't match");
    }
    Matrix S = new Matrix(A.HEIGHT, B.WIDTH)
    for (int i = 0; i < B.WIDTH; i++) {
        for (int j = 0; j < B.HEIGHT; j++) {
            for (int k = 0; k < A.HEIGHT; k++) {
                S.setValue(i, j) += A.getValue(i, k) * B.getValue(k, j);
            }
        }
    }
    return S;
}
```

Figura 4.68 Implementación del método Multiplicación de matrices

Fuente: Elaboración propia (2018)

Clase Imagen (Image)

La siguiente clase, representa lógicamente a la grilla de píxeles (Imagen) que se muestran en los puzles de transformaciones geométricas. Esta clase está compuesta por una matriz de píxeles (Boxes). Esta se define por sus dimensiones (height y width) y comprende los métodos que permiten aplicar transformaciones geométricas sobre imágenes por compresión de dos dimensiones y mostrar gráficamente cada uno de los píxeles en las escenas. En la Figura 4.69 que se muestra a continuación, se presenta la interfaz de esta matriz.

```
public class Image {
    //Attributes
    public Matrix<Box> image;
    public Point[] vertices;
    // Methods
    public Image (int _height, int _width){...
    }
    //Geometric Trasnformations

    private bool pointInTriangle(Point point) {...
    }
    public void applyTraslation(int value,char axis){...
    }
    public void applyEscalation(int value){...
    }
    //Graphic helpers
    public void reDrawing(){...
    }
    public int[,] GetBinaryPatterns(){...
    }
    public void setColor(int x,int y ,Color other){...
    }
    //Getters & Setters
    public int HEIGHT{...
    }
    public int WIDTH{...
    }
}
```

Figura 4.69 Implementación de la clase Imagen (Image)

Fuente: Elaboración propia (2018)

La interfaz de esta clase se divide en 2 secciones importantes: Transformaciones geométricas y Gestión de gráficos.

Dado que el objeto *Matrix<box> image* representa una imagen por compresión de dos dimensiones en un rango de dos colores (Blanco y Negro), se definió una lista de puntos (Point[] vertices) que permite agrupar poligonalmente el área de la imagen visible (píxeles negros), como se puede observar en la Figura 4.70.

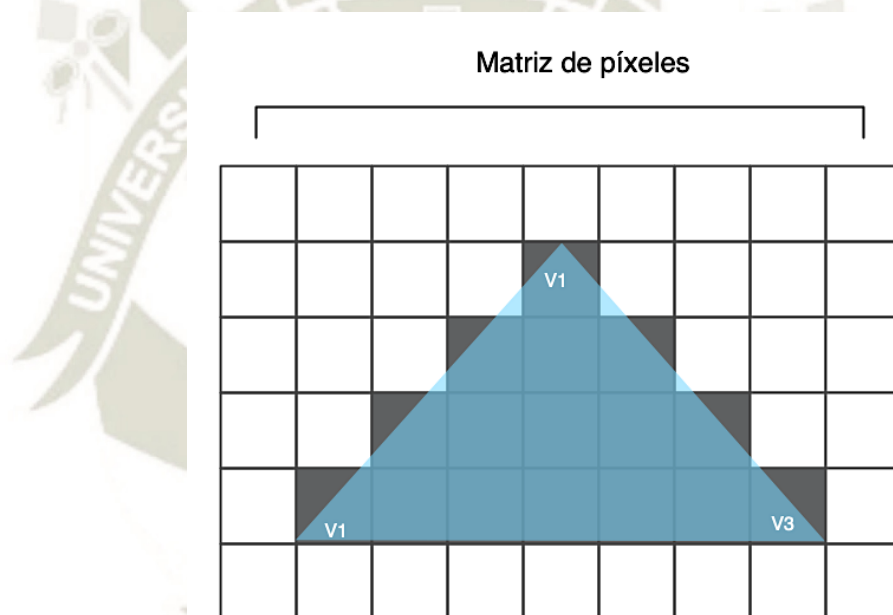


Figura 4.70 Referencia poligonal del área de una imagen

Fuente: Elaboración propia (2018)

Este mecanismo permite trabajar directamente con un polígono al momento de aplicar alguna transformación geométrica. El cual, una vez transformado se vuelve a graficar en píxeles en el plano de la imagen. De esta manera se mantiene una consistencia gráfica con la imagen original.

A continuación, se detallan los métodos más relevantes para esta clase (Image):

a) Punto en un triángulo (PointInTriangle)

Este método permite determinar si un punto $(x,y,1)$ forma parte del área de un triángulo de un plano.

```
private bool pointInTriangle(Point point) {  
    long s1 = abs(vertices[0].cross(vertices[1], vertices[2]));  
    long s2 = abs(point.cross(vertices[0], vertices[1])) +  
             abs(point.cross(vertices[1], vertices[2])) +  
             abs(point.cross(vertices[2], vertices[0]));  
    return s1 == s2;  
}
```

Figura 4.71 Implementación del método `pointInTriangle`

Fuente: Elaboración propia (2018)

Para comprobar que un punto P' se encuentra entre P_1 y P_n (lista de vértices de un polígono) se realiza verificando que el producto cruzado $(P_1 - P_0) \times (P' - P_0)$ es igual a cero o contiene el mismo signo con $(P_1 - P_0) \times (P_n - P_0)$ y $(P_n - P_0) \times (P' - P_0)$ es igual a cero o tiene el mismo signo con $(P_n - P_0) \times (P_1 - P_0)$.

b) Aplicar Traslación (ApplyTraslation)

El siguiente método permite aplicar una traslación geométrica a la imagen ya sea en el eje x o y . Como se explicó anteriormente, la transformación se aplica a un polígono, el cual representa el área visible de la imagen. Una vez que el polígono se ha transformado se vuelve a graficar la imagen en píxeles.

La traslación se aplica en base a las coordenadas homogéneas para una transformación lineal (descrita en la sección de aspectos disciplinarios del presente trabajo). A continuación, se detalla la implementación de este método.

```
public void applyTraslation(int value,char axis){
    // Initial Matrix (P)
    Matrix<int> P = new Matrix<int>(3,this.vertices.Length);
    // Transformation Matrix (T): an identity matrix (3x3)
    Matrix<int> T = new Matrix<int>(3,3,true);
    // Set values to Transformation Matrix
    // Set the translation values (dx & dy)
    // regarding the axis x or y
    int dx = axis=='x'?value:0;
    int dy = axis=='y'?value:0;
    T.SetValue(0,2,dx);
    T.SetValue(1,2,dy);
    // Set values to the Initial Matrix
    // Looping to total number of vertices (3)
    for (int i = 0; i <= this.vertices.Length; i++) {
        // Put the value for each point (x,y,1)
        P.SetValue(i,0,this.vertices[i].point.x);
        P.SetValue(i,1,this.vertices[i].point.y);
        P.SetValue(i,2,1);
    }
    // Transformed Matrix (PT)
    Matrix<int> PT = Matrix.Mult(P,T);
    for (int i = 0; i <= PT.rows; i++) {
        // Put the value for each point (x,y,1)
        Point ptd = new Point(
            PT.GetValue(i,0),
            PT.GetValue(i,1));
        this.vertices[i] = ptd;
    }
}
```

Figura 4.72 Implementación del método *applyTraslation*

Fuente: Elaboración propia (2018)

Como se puede observar en la Figura 4.72, inicialmente se instancia la matriz inicial P, la cual se carga con todos los puntos (x,y,1) del polígono a transformar. Posteriormente, se define la matriz de transformación T, la cual se inicializa como una matriz de identidad, y según a la base teórica de coordenadas geométricas, en la posición [0,2] se establece el valor de transformación dx para el eje x , y en la posición [1,2] el valor de transformación dy para el eje y . Para realizar la traslación se realiza una multiplicación entre las matrices (P y T), obteniendo

como resultado la matriz PT. Esta matriz resultante contiene los nuevos vértices del polígono ya transformado.

c) Aplicar Escalamiento (ApplyEscalation)

Este método permite realizar un escalamiento homogéneo con respecto al origen de una imagen. Para ello, se hace el uso de coordenadas homogéneas para escalamiento geométrico (detallado en la sección de aspectos disciplinarios del presente documento).

```
public void applyEscalation(int value){
    // Initial Matrix (P)
    Matrix<int> P = new Matrix<int>(3,this.vertices.Length);
    // Transformation Matrix (T): an identity matrix (3x3)
    Matrix<int> T = new Matrix<int>(3,3,true);
    // Set the same value to sx & sy
    T.setValue(0,0,value);
    T.setValue(1,1,value);
    // Set values to the Initial Matrix
    // Looping to total number of vertices (3)
    for (int i = 0; i <= this.vertices.Length; i++) {
        // Put the value for each point (x,y,1)
        P.setValue(i,0,this.vertices[i].point.x);
        P.setValue(i,1,this.vertices[i].point.y);
        P.setValue(i,2,1);
    }
    // Transformed Matrix (PT)
    Matrix<int> PT = Matrix.Mult(P,T);
    for (int i = 0; i <= PT.rows; i++) {
        // Put the value for each point (x,y,1)
        Point ptd = new Point(
            PT.getValue(i,0),
            PT.getValue(i,1));
        this.vertices[i] = ptd;
    }
}
```

Figura 4.73 Implementación del método applyEscalation

Fuente: Elaboración propia (2018)

Como se puede observar, el procedimiento es similar al de traslación geométrica visto anteriormente. La diferencia se evidencia, en la inicialización de la matriz de transformación T , donde los valores de transformación (s_x , s_y) se definen en las posiciones $[0,0]$ y $[1,1]$ respectivamente.

Dado que, en los puzzles de escalamiento se expresa específicamente a los estudiantes realizar un escalamiento homogéneo (escalamiento tanto en el eje x como el eje y), en algoritmo se define en base a la igualdad de estos factores [$s_x = s_y$]. El resultado de multiplicación de las matrices P y T , similar al de traslación geométrica, determina los nuevos puntos para el polígono de la imagen.

d) Graficar (ReDrawing)

Este método se ejecuta normalmente después de una transformación geométrica, permite sincronizar el polígono transformado con la matriz de píxeles.

```
public void reDrawing(){
    for (int i = 0; i <= this.image.rows; i++) {
        for (int j = 0; j <= this.image.columns; i++) {
            Box pixel = this.image.getValue(i,j);
            if (this.pointInTriangle(pixel.CENTER)){
                pixel.COLOR = Color.Black;
            }
            else{
                pixel.COLOR = Color.White;
            }
            // replace with transformed pixel
            this.image.setValue(i,j,pixel)
        }
    }
}
```

Figura 4.74 Implementación del método reDrawing

Fuente: Elaboración propia (2018)

Como se puede observar en la Figura 4.74, se hace uso del método *pointInTriangle* descrito anteriormente, para determinar si el centro de cada píxel de la imagen se vio comprometido dentro del área del polígono transformado.

e) Obtener Matriz Binaria (GetBinaryPatterns)

Este método permite obtener una matriz con los valores binarios de la imagen. Se utiliza para facilitar la inicialización de una imagen en el escenario de juego.

```
public int[,] GetBinaryPatterns(){  
    // A new integer matrix is created with the same dimensions  
    int [,] patterns = new int[height,width];  
    // Next, the new matrix is filled with the graphic matrix values  
    for (int i = 0; i < this.height; i++) {  
        for (int j = 0; j < this.width; j++)  
            patterns [i, j] = this.rows [i, j].NUMBER;  
    }  
    return patterns;  
}
```

Figura 4.75 Implementación del método GetBinaryPatterns

Fuente: Elaboración propia (2018)

Como se puede visualizar en la Figura 4.75 el procedimiento es simple, inicialmente se recorre la matriz principal de forma iterativa y se copia directamente los valores a una nueva matriz binaria, la cual es retornada al finalizar los ciclos de iteración.

4.5.2. Motor de bloques y TDL

Este componente agrupa las clases y scripts necesarios para crear casos de prueba, diseñar algoritmos con bloques (Scratch) y ejecutar estos algoritmos y validarlos con los casos de prueba.

A continuación, se describe la implementación de cada uno de estos elementos de acuerdo con la vista lógica planteada en la etapa de diseño de software.

Clase Caso de Prueba (Test Case)

Clase que define la estructura para instanciar casos de prueba dentro del juego. Dado que el objetivo del juego solo comprende operaciones sobre matrices, la interfaz de esta matriz esta particularmente diseñada para evaluar estos procedimientos.

```
public class TestCase () {  
    //Attributes  
    public Matrix<int> A;  
    public Matrix<int> B;  
    public Matrix<int> R;  
    private string operation;  
    private string status;  
    // Methods  
    public TestCase(string operation){ ...  
    }  
    public Check(){ ...  
    }  
    public string STATUS{ ...  
    }  
}
```

Figura 4.76 Interfaz de la clase Caso de Prueba (TestCase)

Fuente: Elaboración propia (2018)

a) Validar Caso de Prueba (Check)

Método que valida el caso de prueba sobre una operación matricial determinada (Suma o Multiplicación).

```
public Check(){
    // Solution Matrix
    Matrix<int> S;
    switch(this.operation){
        case 'suma':
            S = Matrix.Sum(this.A, this.B);
            break;
        case 'multiplicacion':
            S = Matrix.Mult(this.A, this.B);
            break;
    }
    // Verify if calculation is correct
    return this.R == S;
}
```

Figura 4.77 Implementación del método *TestCase.Check*

Fuente: Elaboración propia (2018)

Dado el tipo de operación (Suma o multiplicación), este método evalúa que el cálculo sobre las matrices A y B sea correcto.

Clase Algoritmo (Algorithm)

```
public class Algorithm {
    //Attributes
    public TestCase[] testcases;
    public List<Action> methods;
    private int current;
    // Methods
    public Algorithm(){...}
}
public addMethod(Action action){...}
public Run(){...}
public string CURRENT{...}
}
```

Figura 4.78 Interfaz de la clase *Algorithm (Algorithm)*

Fuente: Elaboración propia (2018)

Clase que comprende la estructura lógica para evaluar el diseño de un algoritmo. Agrupa una lista de funciones (List<Actions>) que definen un comportamiento variado. Además, contiene un arreglo de casos de prueba (TestCases) los cuales permiten validar las funciones de la lista de acciones cuyo índice *current* determina la acción seleccionada.

a) Añadir método (addMethod)

Método que permite añadir una función a la lista de acciones (methods).

b) Ejecutar (Run)

Ejecuta un método de la lista de acciones, el cual es determinado por el índice *current*, este método se agrupa dentro de un bloque de manejo de excepciones (try-catch). En caso de una falla, se retorna el mensaje de error que se utiliza como retroalimentación en la escena correspondiente.

```
public Run(){
    try{
        // Run an specify method with undefined test cases
        this.methods[this.current].Invoke(this.testcases);
    }catch(Exception e){
        return e.getMessage();
    }
}
```

Figura 4.79 Implementación del método Algorithm.Run

Fuente: Elaboración propia (2018)

4.5.3. Acceso a datos (Data Access)

Este componente comprende todos los elementos necesarios que permiten la comunicación entre el cliente (Juego) y servidor (API).

Clase IConnector

Permite realizar una petición rápida para recuperar un contenido desde una URL específica. Se hace uso del módulo de Unity (UnityEngine:WWW) para realizar una

comunicación HTTP con un servidor externo. A continuación, se detalla los métodos más relevantes de la implementación de la clase.

```
public class IConnector {  
    // Attributes  
    private string url ;  
    private WWW www;  
    public string response;  
    // Methods  
    public IConnector(string url){  
        this.url = url  
        this.www = new WWW(url);  
    }  
    public void Dispatch (MonoBehaviour controller) {--  
    }  
    private IEnumerator WaitForRequest(){ ...  
    }  
    public string RESPONSE{--  
    }  
}
```

Figura 4.80 Interfaz de la clase IConnector

Fuente: Elaboración propia (2018)

a) Esperar petición (WaitForRequest)

Método de la clase *IConnector* que permite ejecutar una instrucción de rendimiento (*yield*) para suspender una subrutina de petición a un servidor (www) y retomarlo en el siguiente fotograma. Se utiliza para ejecutar tareas que toman tiempo de ejecución y no detener el juego. Al finalizar la instrucción de rendimiento *yield* se devuelve la respuesta de la petición en caso de no presentar algún error en el servidor.

```
private IEnumerator WaitForRequest(){
    yield return this.www;
    try{
        if (!this.www.error){
            this.response = this.www.data;
        } else {
            throw new Exception("Request has failed: "+ this.www.error);
        }
    }catch(Exception e){
        Debug.Log(e.getMessage());
    }
}
```

Figura 4.81 Implementación del método *IConnector.WaitForRequest*

Fuente: Elaboración propia (2018)

b) Enviar Solicitud (Dispatch)

Este método inicia una subrutina *WaitForRequest* a través del método *StartCoroutine* de la clase *Monobehaviour*.

```
public void Dispatch (MonoBehaviour controller) {
    controller.StartCoroutine(this.WaitForRequest());
}
```

Figura 4.82 Implementación del método *IConnector.Dispatch*

Fuente: Elaboración propia (2018)

4.5.4. Control de seguridad (Copyright)

Según la especificación de requerimientos, es necesario que el juego pueda ejecutarse en ambientes dentro del campo de experimentación (Laboratorios de la universidad). Por lo que se realiza una petición (utilizando el componente de Acceso a Datos: *IConnector*) para validar que la URL del cliente pertenezca a las permitidas de la universidad). Esta acción se ejecuta por única vez al iniciar el juego por lo que en caso de que la IP no sea permitida, el usuario será redirigido a una escena de derechos de autor (*CopyrightScene*).

```
public class Copyright : MonoBehaviour {  
  
    public GameObject play_scene;  
    // Use this for initialization  
  
    public static string LocalIPAddress(){--  
    }  
    void Start () {  
        IConnector connector =  
        new IConnector('api.enginners-escaping.test/validate/'+  
        | | | | | this.LocalIPAddress());  
  
        connector.Dispatch(this);  
        if(!connector.RESPONSE){  
            Application.LoadLevel ("CopyrightScene");  
        }  
        else{  
            play_scene.SetActive (true);  
        }  
    }  
}
```

Figura 4.83 Implementación del Controlador de seguridad (Copyright)

Fuente: Elaboración propia (2018)

4.5.5. Controladores de personaje (Scripts)

Los controladores son aquellos scripts que se asocian directamente al objeto de juego, definen el comportamiento de este en cada fotograma en tiempo de ejecución. Estos se implementan a través de la clase *MonoBehaviour* lo cual permite el acceso a otros componentes estándar de Unity.

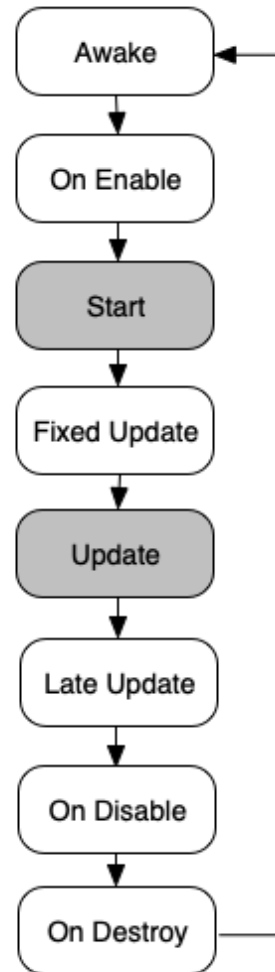


Figura 4.84 Ciclo de vida de un controlador de juego (Monobehaviour)

Fuente: Elaboración propia (2018)

Como se puede observar en la Figura 4.84, los controladores de juego se rigen sobre un ciclo de vida que se basa en una lista de estados [Awake, On Enable, Start, Fixed Update, Update, LateUpdate, On Disable, On Destroy] los cuales son necesarios para la construcción de un controlador.

Dentro los más relevantes se tiene: a) El método **Start()**: El cual se ejecuta una sola vez permite inicializar un componente de juego, b) El método **Update()**: Es la más usada, se llama por cada fotograma y se ejecuta de manera constante, permite establecer el comportamiento del objeto de juego, c) El método **OnDestroy()**: Se

ejecuta cuanto el objeto de juego es destruido y **FixedUpdate()**: El cual tiene un comportamiento similar al método *Update*, sin embargo, se ejecuta a diferente velocidad (0,02 segundos aproximadamente) de manera independiente para realizar los cálculos físicos.

El script o controlador de personaje (*CharacterController*) define el comportamiento del personaje principal, comprende las acciones de movimiento en primera persona (Saltar, correr y caminar) denominado también como la mecánica de juego.

```
public class CharacterController : MonoBehaviour {
    /* PUBLIC VARIABLES */
    public string name = "";
    public float jumpForce = 0f;
    public float velocity = 0f;
    public Collider2D collider;
    private Rigidbody2D rigidBody;
    private Animator animator;
    private GameObject ground_detector;
    /* STATIC VARS */
    public static GameObject player;
    public static bool inRange = false;
    // Use this for initialization
    void Start () {
    }
    // Update is called once per frame
    void Update () {
    }
    void OnAction(){
    }
    void FixedUpdate(){
    }
    public void ResetCharacter(){
    }
    public void SetPosition(){
    }
    void OnDead(){
    }
    void Run (){
    }
}
```

Figura 4.85 Interfaz del controlador del personaje principal

Fuente: Elaboración propia (2018)

Además, se establecen los parámetros de animación que permiten gestionar y direccionar las animaciones. Finalmente se define la física y mecánica de colisión del jugador con su entorno. Los atributos públicos en un controlador o script permiten asociar directamente componentes al objeto de juego, así mismo, también permite definir propiedades de ámbito público que determinan su comportamiento desde el escenario de juego.

El controlador del personaje se definió en base a los siguientes elementos: a) **Animator**: Permite gestionar las animaciones, b) **Rigidbody2D**: Integra la física al personaje, c) **Collider**: Define el área de colisión, d) **jumpForce** y **velocity**: Permiten gestionar la magnitud de las fuerzas que determinan la fuerza de salto y velocidad del personaje, y e) **ground_detector**: Elemento que almacena una referencia del objeto que colisiona con el suelo escenario.

Como se explicó anteriormente, en el método **Start()** se definen y se cargan los elementos necesarios para inicializar un objeto de juego de un solo comportamiento.

```
void Start () {  
    // Get Rigidbody2d from GameObject  
    rigidBody = GetComponent<Rigidbody2D> ();  
    // Get Animator from GameObject  
    animator = GetComponent<Animator> ();  
    // Get the ground dector from GameObject Children  
    foreach (Transform obj in GetComponentsInChildren<Transform> ()) {  
        if (obj.name == "detectorSuelo") {  
            ground_detector = obj;  
            break;  
        }  
    }  
    // Save a reference of the player  
    ProjectVars.Instance.player = this.gameObject;  
    // Register available actions as public notifications  
    NotificationCenter.DefaultCenter().AddObserver(this,  
    ["OnDead", "StopAutoRun", "AutoRun", "Block", "UnLock", "SetPosition"]);  
}
```

Figura 4.86 Inicialización del personaje principal

Fuente: Elaboración propia (2018)

Así mismo como se puede observar en la Figura 4.86, se define una variable global (*ProjectVars.Instance.Player*), el cual guarda una referencia del *gameObject* del objeto Jugador, con la finalidad de permitir su acceso desde cualquier parte del juego. Dado que las acciones del jugador se gestionan mediante notificaciones en la última parte de este método se registran todas las acciones que el jugador puede realizar de manera independiente.

```
void Update () {
    float velx = 0f;
    if (isDead )
        return;
    // turn right
    if (Input.GetKey (KeyCode.RightArrow)|| Input.GetKey (KeyCode.D)){
        velx = Mathf.Abs (velocity);
        orientation = -1f;
    }
    // turn left
    if ( (Input.GetKey (KeyCode.LeftArrow)|| Input.GetKey (KeyCode.A))){
        velx = velocity * -1f;
        orientation = 1f;
    }
    //Jump
    if (Input.GetKey (KeyCode.Space) && enSuelo && !jumpbutton){
        rigidBody.velocity = new Vector2 (rigidBody.velocity.x, jumpForce);
    }
    // Enter in a room
    if (Input.GetKey (KeyCode.UpArrow)|| Input.GetKey (KeyCode.W) ){
        EnterTheDoor();
    }
    // Show a puzzle
    if (Input.GetKey (KeyCode.E)){
        OnAction ();
    }
    // Set velocity param
    this.rigidBody.velocity = new Vector2 (velx, rigidBody.velocity.y);
    // Set animator value
    this.animator.SetBool ("isRun", velx != 0f);
}
```

Figura 4.87 Implementación de la mecánica del jugador principal

Fuente: Elaboración propia (2018)

El método *Update* se llama por cada fotograma del juego, su ejecución es de forma recurrente, por lo tanto, es el espacio idóneo para capturar los eventos de los dispositivos de entrada (ratón o teclado).

Como se puede ver en la Figura 4.87, para el caso del controlador del personaje principal. En este método se establece la mecánica del jugador (caminar, girar a la derecha o izquierda, saltar, entrar a una habitación y resolver un puzle) a través de los eventos del teclado, además se regula la velocidad y dirección del jugador a través de vectores de movimiento y se controla los parámetros de animación.

El método *FixedUpdate()* a diferencia del método *Update()* se ejecuta en un espacio de fotogramas determinado (frecuencia fija) después de realizar los cálculos de física.

```
void FixedUpdate(){  
    // Get Ground LayerMask  
    mascaraSuelo = LayerMask.GetMask("Ground");  
    enSuelo = Physics2D.OverlapCircle(  
        comprobadorSuelo.position,  
        comprobadorRadio, mascaraSuelo);  
    animator.SetBool ("inGround", enSuelo);  
}
```

Figura 4.88 Implementación de la física (Gravedad) del personaje principal

Fuente: Elaboración propia (2018)

En el método *FixedUpdate* del personaje principal, se define el parámetro de animación “In Gound” tras verificar una colisión entre el jugador y el suelo del escenario el cual se determina en base la física calculada (Gravedad).

4.5.6. Generador de niveles (Level generator)

El generador de niveles facilita la creación del escenario de juego. Este componente no se desarrolló inicialmente como parte de la primera versión, sin

embargo, se introdujo en el desarrollo de la versión final del juego. Este componente solo comprende una clase: *PlatformController*, el cual se implementa de la clase base *MonoBehaviour* como un script o controlador de juego.

```
public class PlatformGenerator : MonoBehaviour {  
  
    private Rigidbody2D rb;  
    public GameObject[] objects;  
    public float time = 3f;  
    // Use this for initialization  
    void Start () {  
    }  
    void Instance(){  
    }  
}
```

Figura 4.89 Interfaz de la clase *PlatformGenerator* (Generador de niveles)

Fuente: Elaboración propia (2018)

La lógica de ese componente es simple, dada una lista de bloques o grupo de objetos anidados pre-construidos, estos se instancian de forma aleatoria (no repetida) en línea horizontal según un tiempo determinado.

```
void Instance(){  
    int index = Random.Range (0, objects.Length);  
    // To avoid re instiate the same block  
    if (ProjectVars.Instance.actual_block != "") {  
        index = int.Parse (ProjectVars.Instance.actual_block);  
    }  
    // Instantiate a pre building group of objects  
    GameObject block = Instantiate(  
        objects [index],  
        this.transform.position,  
        objects [index].transform.rotation  
    );  
}
```

Figura 4.90 Método de instanciación aleatoria (Generador de niveles)

Fuente: Elaboración propia (2018)

De acuerdo con la dimensión de cada uno de estos bloques y para evitar que se solapen, se estableció un tiempo de instanciación constante de 3 segundos.

4.6. Construcción de las escenas

Esta sección detalla el procedimiento para la construcción de los objetos de juego (GameObjects), escenas, animaciones del Juego Serio y la integración con los scripts o controladores de la sección anterior.

4.6.1. Creación de Objetos de juego

Un Objeto de juego o *GameObject* es un contenedor de componentes. Los objetos de juego en unity generalmente se construyen en base a un *Prefab*. Un *Prefab* permite almacenar un objeto (*GameObject*) es su totalidad, con sus componentes, gráficos, animaciones y propiedades. Por lo tanto, un *Prefab* actúa como plantilla, facilitando la instanciación de objetos en las escenas.

a) Personaje principal (Prefab)

El personaje principal se compone de manera articulada distribuida en sus partes, con la finalidad de facilitar su animación y establecer específicamente un área de colisión óptimo.

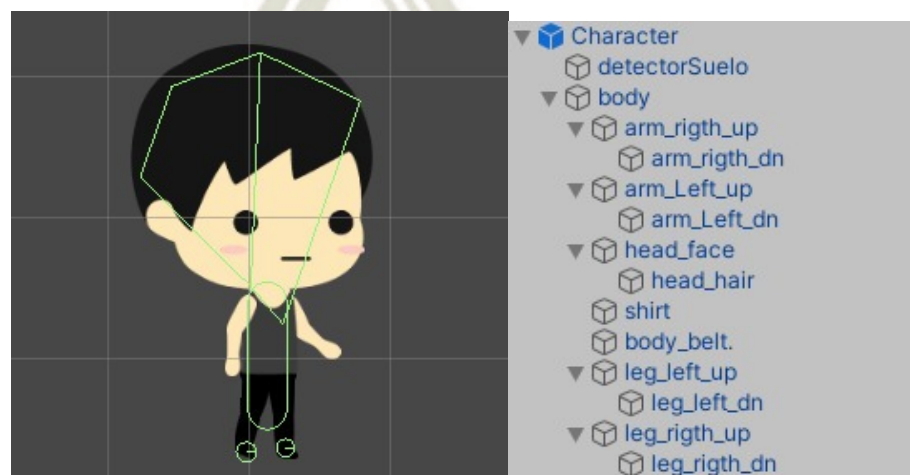


Figura 4.91 Prefab del personaje principal

Fuente: Elaboración propia (2018)

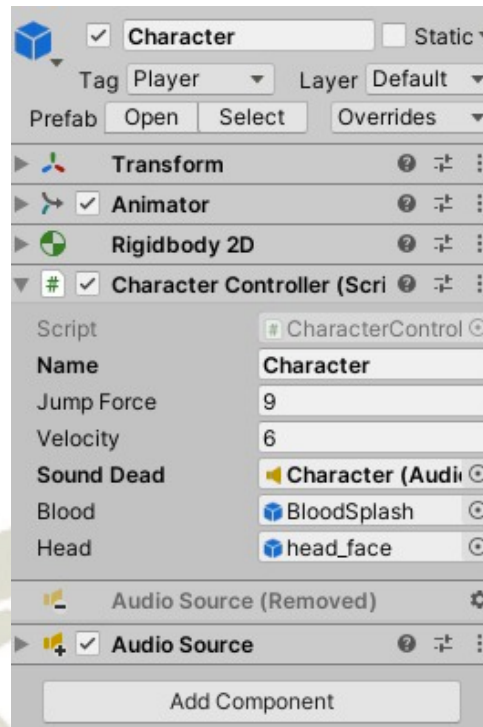


Figura 4.92 Componentes adjuntos del personaje principal

Fuente: Elaboración propia (2018)

El área de colisión como se ve en la Figura 4.91 Prefab del personaje principal está compuesta por polígonos y se define a través de un *Collider* (Componente de Unity para este fin). Por lo general, es una buena práctica utilizar figuras circulares para permitir al jugador escalar sobre las plataformas del juego y evitar algún posible atasco durante el recorrido.

Los componentes en Unity son piezas funcionales que determinan el comportamiento de un objeto de juego (*GameObject*). Por lo general, un objeto inicia con un componente por defecto (*Transform*) el cual determina su posición en el espacio.

Para la construcción del personaje principal, se adjuntó un componente *RigidBody2D* que integra la física sobre el objeto. Además, se adjuntó un componente *Animator*, que permite administrar las animaciones y finalmente un componente *Script* (*CharacterController*) el cual define las acciones y propiedades del objeto.

b) Trampas y Obstáculos

Las trampas y obstáculos tienen una composición simple, presentan un área de colisión geométrica. Dada su naturaleza cada una de ellas se almacena en un *Prefab* independiente para poder reutilizar en la creación de las escenas.

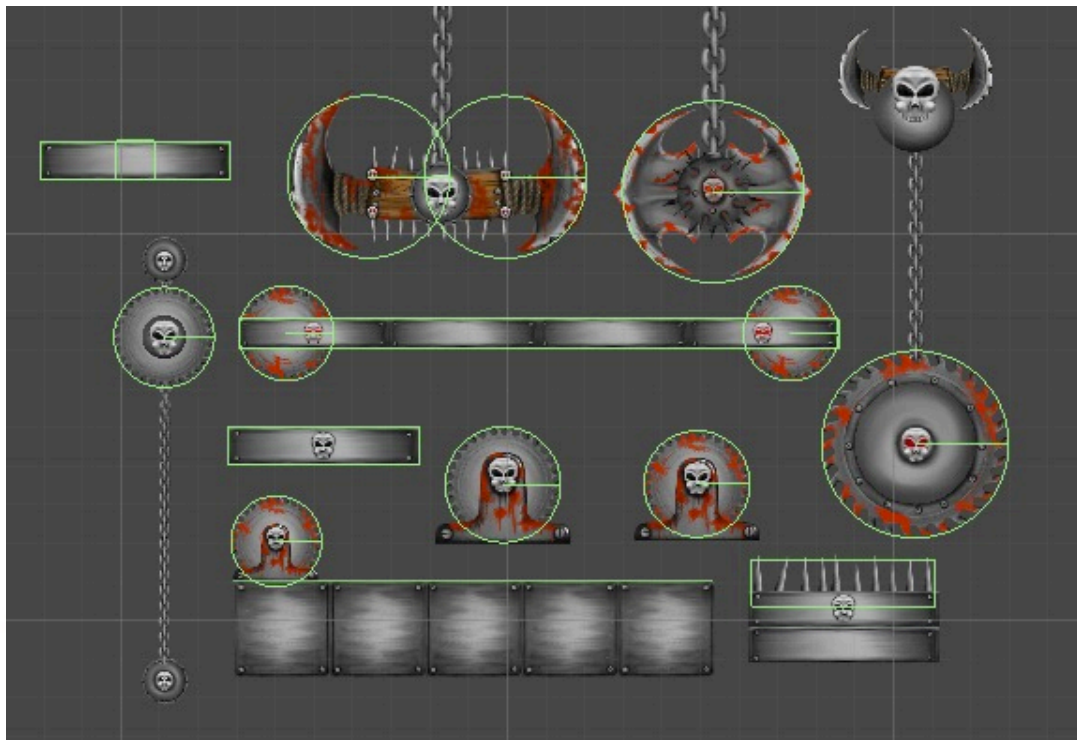


Figura 4.93 Componentes preconstruidos (*Prefabs*) trampas y obstáculos

Fuente: Elaboración propia (2018)

Los componentes asociados en estos objetos por lo general son del tipo *Animator* que permite añadir animaciones perpetuas y *AudioSource* para adjuntar un sonido específico.

4.6.2. Proceso de animación

Unity tiene un sistema de animación llamado *Mecanim*, el cual proporciona un flujo de trabajo de animaciones simple para alinear clips de animación para cualquier objeto en escena.

Las animaciones se definen en máquinas de estado simple, como se puede ver en la Figura 4.94 como se muestra a continuación.

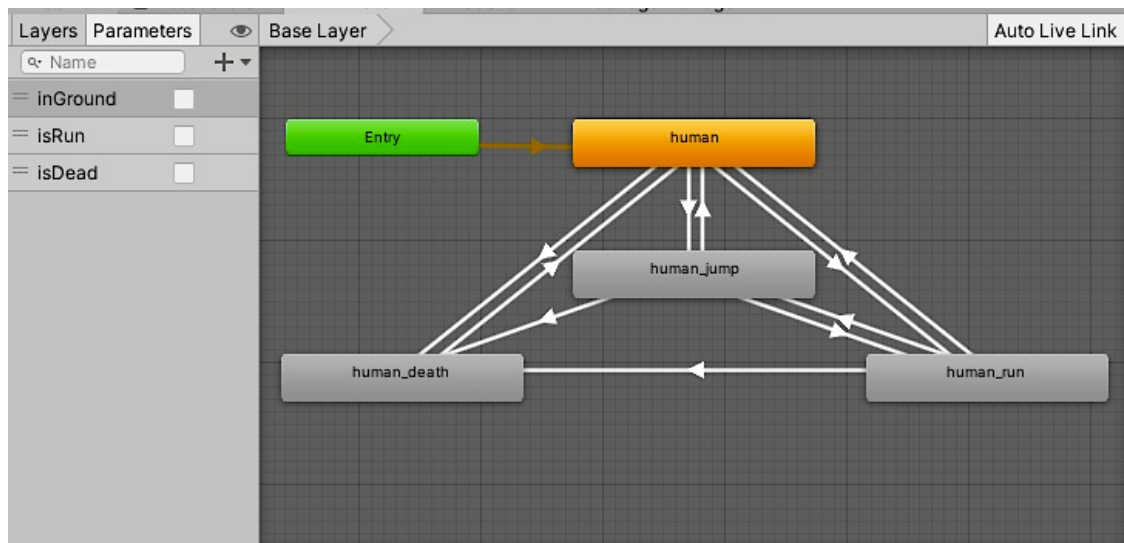


Figura 4.94 Máquina de estado finito de animación del personaje principal

Fuente: Elaboración propia (2018)

Los clips de animación para el personaje principal fueron desarrollados desde el mismo editor de animaciones que proporciona Unity. Estos clips se adjuntan a un componente *Animator Controller*, previamente asociado al objeto que se desea animar (Personaje Principal). Como se puede observar en la figura anterior, los clips de animación se pueden visualizar como estados desde la ventana de *Animator*. Del lado izquierdo de esta ventana se definen los parámetros de la máquina de estados, los cuales pueden ser booleanos, numéricos y flotantes. Estos parámetros, permiten el cambio de estado entre los nodos de animación y pueden modificarse desde cualquier parte del juego.

4.6.3. Creación de escenas

De acuerdo con el diseño de niveles de la etapa de diseño y planificación, se estableció implementar 4 niveles o capítulos con diferente dificultad en los puzzles o juegos de lógica de resolución de problemas de matrices. A continuación, se muestran de manera general las escenas y la distribución de sus elementos para cada uno de ellos.

a) Nivel Introductorio

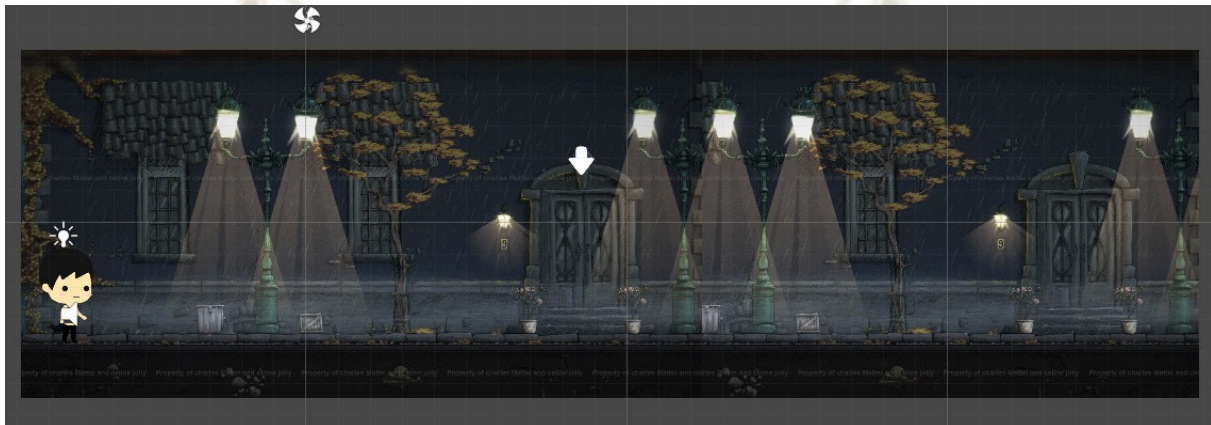


Figura 4.95 Escenario del nivel introductorio

Fuente: Elaboración propia (2018)

b) Primer nivel

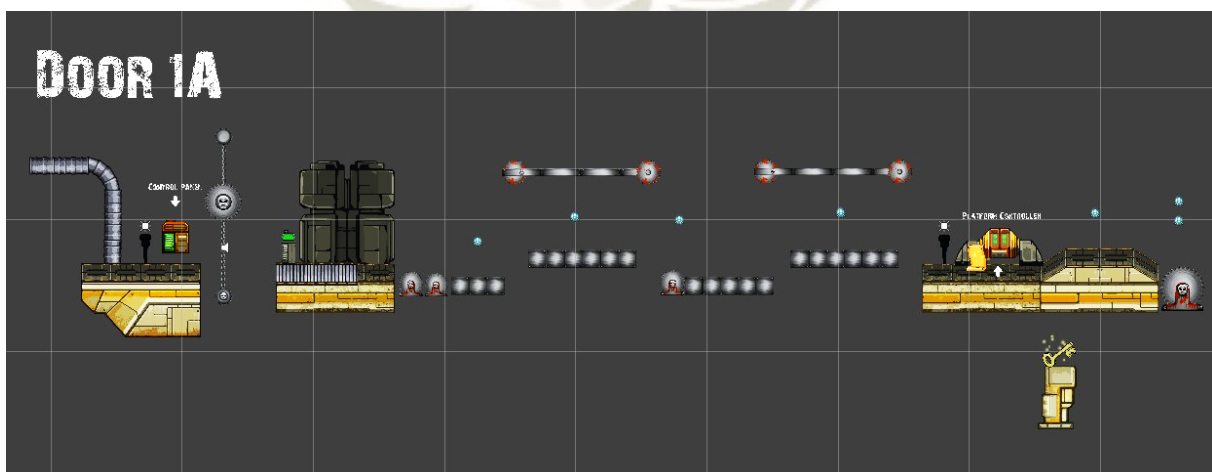


Figura 4.96 Escenario del primer nivel

Fuente: Elaboración propia (2018)

c) Segundo nivel

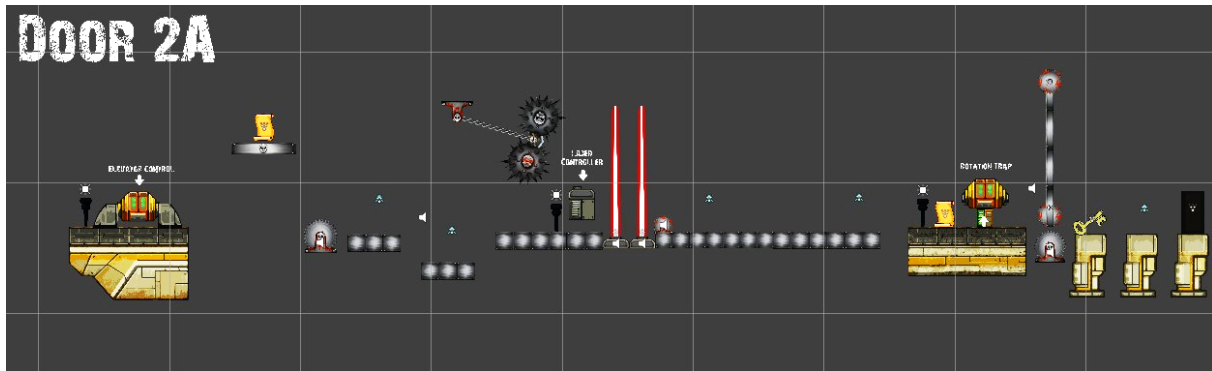


Figura 4.97 Escenario del segundo nivel

Fuente: Elaboración propia (2018)

d) Tercer nivel



Figura 4.98 Escenario del tercer nivel

Fuente: Elaboración propia (2018)

e) Cuarto nivel

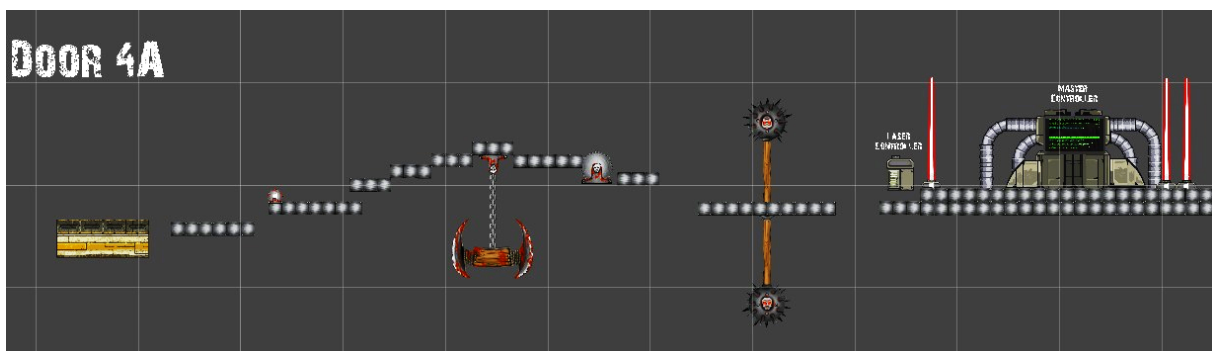


Figura 4.99 Escenario del cuarto nivel

Fuente: Elaboración propia (2018)

f) Hall o recepción (Todos los niveles)

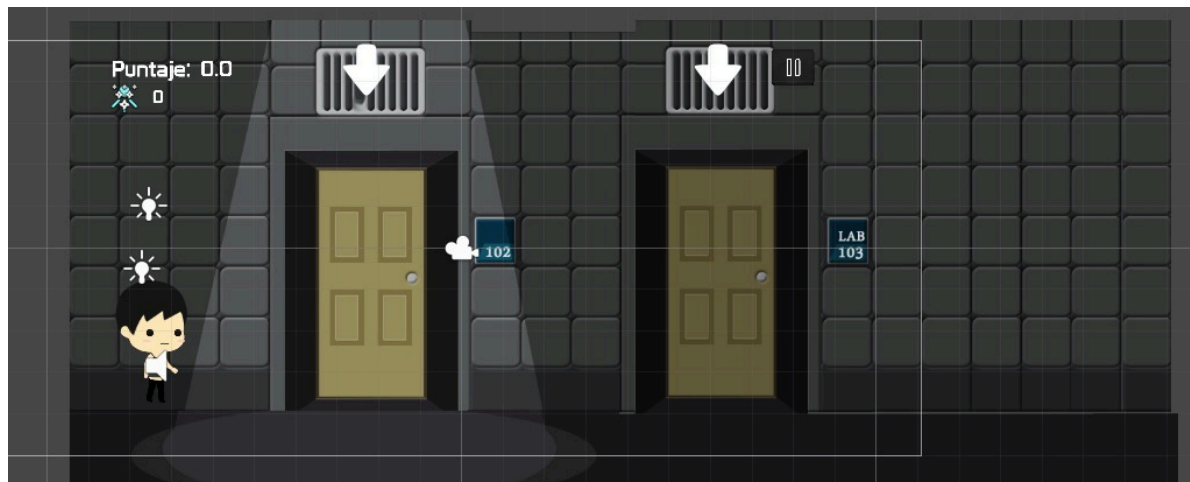


Figura 4.100 Escenario del Hall o Recepción

Fuente: Elaboración propia (2018)

4.7. Página web de presentación

Dado que la ejecución juego se tenía que desarrollar en un total de 2h como máximo, se implementó una página web simple que permitiese descargar el juego (Windows / OSX) y/o jugar en línea (Web Gl). Su utilización facilitó la instalación del juego dentro de los 10 primeros minutos de la clase.

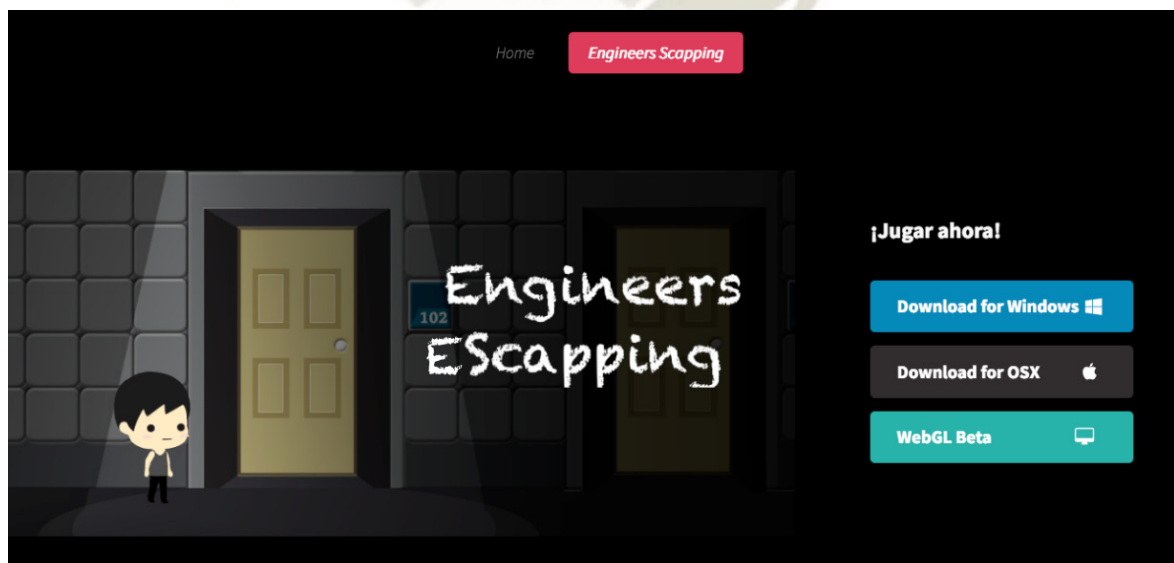


Figura 4.101 Página web de presentación

Fuente: Elaboración propia (2018)

5. Pruebas

En esta sección se definen las pruebas de software realizadas al producto del presente trabajo de investigación. Estas se ejecutaron en la etapa de validación de la metodología SCRUM al finalizar cada Sprint.

5.1. Objetivo

El objetivo de este apartado es definir los casos de pruebas para verificar que el Juego Serio satisface los requisitos funcionales. La etapa de pruebas comprende la definición de los casos de prueba, la trazabilidad entre casos de pruebas y requerimientos, y la estrategia para la ejecución de dichas pruebas.

5.2. Alcance

El alcance o cobertura de pruebas se acotó en base a los requerimientos funcionales de software que agrupan el contenido académico propuesto y funciones básicas. A continuación, se listan cada uno de ellos.

Tabla 4.38
Alcance de pruebas: Lista de requerimientos

ID	Requerimiento
RE01	Mostrar la pantalla principal para iniciar el juego
RE02	El juego debe mostrar audiovisualmente la historia
RE03	Permitir la autenticación de los usuarios
RE04	El juego debe permitir acceder a los 4 niveles de juego
RE06	Abrir compuertas y cerrojos dentro del juego
RE07	Resolver un juego de lógica sobre representación de imágenes
RE09	Resolver un juego de lógica sobre traslaciones geométricas
RE10	Resolver un juego de lógica sobre rotaciones geométricas
RE11	Resolver un juego de lógica sobre escalamiento geométrico
RE12	Resolver una trivía sobre matrices
RE14	El juego debe acumular el puntaje al momento de resolver juegos de lógica
RE18	Visualizar el ranking general
RE20	Diseñar un algoritmo de suma de matrices en base a TDL

Elaboración propia

5.3. Trazabilidad de las pruebas

En este apartado se presenta matriz como la que se muestra a continuación, en la cuál se detalla la correspondencia relacional entre los casos de pruebas definidos, y los requerimientos funcionales enlistados anteriormente. Las columnas representan a los casos de pruebas definidos, y las filas los requisitos funcionales. Si un caso de prueba se encarga de verificar un requerimiento, se coloca una X en la intersección.

Tabla 4.39
Trazabilidad de pruebas

	CP1	CP2	CP03	CP04	CP05	CP06	CP07	CP08	CP09	CP10
RE01	X								X	
RE02		X								
RE03		X								
RE04		X								
RE06			X	X	X	X	X	X		X
RE07				X						
RE09						X				
RE10							X			
RE11								X		
RE12					X					
RE14					X	X	X	X		X
RE18									X	
RE20										X

Elaboración propia (2018)

5.4. Definición de casos de prueba

En este apartado se describen en detalle cada uno de los casos de pruebas que se hayan identificado como necesarios para verificar la funcionalidad completa del Juego Serio. El conjunto de casos de pruebas que se muestran a continuación fue ejecutado de manera secuencial con la finalidad de asegurar el correcto despliegue de la aplicación.

Tabla 4.40

Caso de prueba 01 – Iniciar el juego

CP01 Iniciar el juego

Realizar una correcta inicialización del juego dentro del área de ejecución permitido

Pre-requisitos

- Verificar que la IP del origen de la ejecución este en la lista de IPs permitidas

Pasos

1. Ejecutar el juego
2. Si el jugador tiene una IP permitida
 - 2.1. Ver la pantalla principal
3. Sino
 - 3.1. Ver una pantalla de bloqueo

Resultado esperado

Si el jugador tiene una IP permitida deberá poder acceder a la pantalla principal, de lo contrario se le mostrará una pantalla de bloqueo.

Elaboración propia

Tabla 4.41

Caso de prueba 02 – Autenticar al usuario

CP02 Autenticar al usuario

Al finalizar la historia del juego se debe ingresar un código de estudiante que autentique al estudiante

Pre-requisitos

- Inicializar el juego con una IP permitida

Pasos

1. Inicializar el juego
2. Visualizar la historia de juego
3. Ingresar el código del estudiante cuando se le solicite
4. Continuar con la lista de niveles

Resultado esperado

Iniciado el juego, se le muestra la historia y se le presenta los objetivos al jugador. Al solicitar la autenticación del usuario el jugador ingresa un código universitario que una vez validado continuará con la selección del capítulo.

Elaboración propia

Tabla 4.42

Caso de prueba 03 – Validar las acciones básicas

CP03 Validar las acciones básicas

Al iniciar el primer nivel con una previa autenticación del estudiante. Se debe validar que el jugador pueda recorrer de forma libre dentro del escenario del primer nivel.

Pre-requisitos

- Autenticación del usuario

Pasos

1. Iniciar el primer nivel.
2. Recorrer el escenario.
3. Saltar y moverse sobre las plataformas.

Resultado esperado

El jugador debe poder desplazarse de forma libre sobre el escenario sin evitar atascos durante el proceso.

Elaboración propia

Tabla 4.43

Caso de prueba 04 – Realizar un puzle de representación de imágenes

CP04 Realizar un puzle de representación de imágenes

Al abrir una compuerta o desactivar una trampa en el primer nivel, se abrirá un juego de lógica (puzle) sobre representación de imágenes que al completarlo se debe continuar en el juego.

Pre-requisitos

- Autenticación del usuario

Pasos

1. Iniciar el primer nivel.
2. Recorrer el escenario.
3. Acercarse a puerta principal
4. Pulsar la tecla “E” para habilitar el puzle.
5. Resolver el puzle dibujando el patrón numérico.
6. Desactivar el obstáculo.

Resultado esperado

El jugador al resolver el puzle de representación de imágenes desactiva el obstáculo y continua con el juego

Elaboración propia (2018)

Tabla 4.44

Caso de prueba 05 – Realizar un puzle de trivia de matrices

CP05 Realizar un puzle de trivia de matrices

Al abrir una compuerta al inicio del segundo nivel se pueden encontrar “paneles de control” asociados a una trivia sobre conceptos de matrices, que al resolverlo se desactiva un obstáculo y permite al jugador continuar en el juego.

Pre-requisitos

- Autenticación del usuario

Pasos

1. Recorrer el escenario.
2. Acercarse a un panel de control de trivia de matrices.
3. Pulsar la tecla “E” para acceder el puzle
4. Resolver la trivia
5. Desactivar el obstáculo.

Resultado esperado

El jugador al resolver el puzle de trivia desactiva el obstáculo y continua con el juego

Elaboración propia

Tabla 4.45

Caso de prueba 06 – Realizar un puzle de traslación geométrica

CP06 Realizar un puzle de traslación geométrica

Al abrir una compuerta al inicio del segundo nivel se pueden encontrar “paneles de control” asociados a un problema sobre traslación geométrica, al resolverlo se desactiva el obstáculo adjunto y permite al jugador continuar en el juego.

Pre-requisitos

- Autenticación del usuario
- Representación de imágenes

Pasos

1. Recorrer el escenario.
2. Acercarse a un panel de control de traslación geométrica.
3. Pulsar la tecla “E” para acceder el puzle
4. Resolver la transformación con coordenadas homogéneas para traslación.
5. Desactivar el obstáculo.
6. Continuar con el juego.

Resultado esperado

El jugador al resolver el puzle de traslación geométrica mueve la imagen propuesta, se desactiva el obstáculo y continua con el juego.

Elaboración propia

Tabla 4.46

Caso de prueba 07 – Realizar un puzle de rotación geométrica

CP07 Realizar un puzle de rotación geométrica

Al abrir una compuerta al inicio del tercer nivel se pueden encontrar “paneles de control” asociados a un problema sobre rotación geométrica, al resolver se desactiva el obstáculo adjunto y permite al jugador continuar en el juego.

Pre-requisitos

- Autenticación del usuario

- Representación de imágenes

Pasos

1. Recorrer el escenario.
2. Acercarse a un panel de control de rotación geométrica.
3. Pulsar la tecla “E” para acceder el puzle
4. Ingresar el ángulo de rotación.
5. Visualizar la transformación

Resultado esperado

El jugador al resolver el puzle de rotación geométrica gira la imagen en un ángulo de inclinación determinado, se desactiva el obstáculo y continua con el juego.

Elaboración propia

Tabla 4.47

Caso de prueba 08 – Realizar un puzle de escalamiento geométrica

CP08 Realizar un puzle de escalamiento geométrico

Al abrir una compuerta al final del tercer nivel y al inicio del cuarto nivel se pueden encontrar “paneles de control” asociados a un problema sobre escalamiento geométrico, al resolverlo se desactiva el obstáculo adjunto y permite al jugador continuar en el juego.

Pre-requisitos

- Autenticación del usuario
- Representación de imágenes
- Traslación geométrica

Pasos

1. Recorrer el escenario.
2. Acercarse a un panel de control de escalamiento geométrico.
3. Pulsar la tecla “E” para acceder el puzle
4. Resolver el escalamiento geométrico a través de coordenadas geométricas.
5. Visualizar la transformación
6. Desactivar el obstáculo.
7. Continuar con el juego.

Resultado esperado

El jugador al resolver el puzle de escalamiento geométrico incrementa el tamaño de la imagen en una cantidad de píxeles determinados, se desactiva el obstáculo y continúa con el juego.

Elaboración propia

Tabla 4.48

Caso de prueba 09 – Visualizar el ranking general

CP09 Visualizar el ranking general

Desde el menú de niveles se debe poder acceder al ranking de puntaje de todos los participantes para cada uno de los capítulos o niveles.

Pre-requisitos

- Autenticación del usuario

Pasos

1. Ingresar al menú principal
2. Ingresar al menú de capítulos (Niveles)
3. Seleccionar la opción “Ranking”
4. Visualizar en una lista el puntaje de todos los participantes

Resultado esperado

El jugador al resolver el puzle de escalamiento geométrico incrementa el tamaño de la imagen en una cantidad de píxeles determinados, se desactiva el obstáculo y continúa con el juego.

Elaboración propia

Tabla 4.49

Caso de prueba 10 – Diseñar un algoritmo sobre matrices (TDL)

CP09 Diseñar un algoritmo sobre matrices utilizando TDL

Al abrir una compuerta al final del cuarto y último nivel se encuentra un único módulo asociado a un problema de diseño de al, al resolverlo se desactiva el obstáculo adjunto y permite al jugador continuar en el juego.

Pre-requisitos

- Autenticación del usuario

Pasos

5. Ingresar al menú principal
6. Ingresar al menú de capítulos (Niveles)
7. Seleccionar la opción “Ranking”
8. Visualizar en una lista el puntaje de todos los participantes

Resultado esperado

El jugador al resolver el puzle de escalamiento geométrico incrementa el tamaño de la imagen en una cantidad de píxeles determinados, se desactiva el obstáculo y continua con el juego.

Elaboración propia

5.5. Pruebas Beta

Se realizaron las pruebas en producción en una versión beta del juego en el grupo 3 de Pensamiento Computacional de primer semestre en el año 2017, donde los participantes aceptaron otorgar sus datos y uso de su imagen para el proyecto de investigación. Estas pruebas permitieron validar los requerimientos no funcionales (Portabilidad, Seguridad y Usabilidad) y se ejecutaron bajo el siguiente cronograma.

Tabla 4.50

Plan de ejecución del Juego Serio

Actividad	Duración estimada (minutos)	Duración Real (minutos)
Presentación de la actividad	15	15
Prueba de entrada (Pre-test)	20	20
Instalación del juego	5	10
Ejecución del juego	60	70
Prueba final (Post-test)	20	20
Total	120	135

Elaboración propia

Inicialmente se indicó en qué consistía la actividad en una presentación de 15 minutos, seguidamente se tomo un examen de entrada (pre-test) con una duración de 20 minutos, con la finalidad de recabar el estado primario de los participantes.

Posteriormente, los estudiantes instalaron el juego donde se realizaron las validaciones del requerimiento no funcional: Portabilidad (sobre un entorno web y de escritorio). Asimismo, se validó que solo los participantes tuvieran acceso al Juego Serio (Seguridad). Para ello se solicitó un acceso temporal al servidor en la oficina de informática de la universidad desde laboratorio donde se realizó la experimentación, de esa forma se pudo verificar el sistema de autenticación implementado.

Finalizada la actividad, los estudiantes fueron sometidos a un examen de conocimientos (post-test) con una duración de 20 minutos para evaluar la efectividad de la propuesta (Juego Serio). La clase terminó con un excedente de 15 minutos del tiempo establecido dado unos problemas de conectividad que presentaron algunos estudiantes.



Figura 4.102 Aplicación del juego en el grupo 3 de Pensamiento Computacional (2017)

Fuente: Elaboración propia (2018)

5.6. Retroalimentación

La retroalimentación recabada la final de la etapa de pruebas beta fue constructiva gracias a los estudiantes que fueron muy abiertos al expresar su punto de vista, esto nos permitió evaluar y validar el requerimiento no funcional: Usabilidad. A continuación, se detallan las mas relevantes:

a) Actividades largas y tiempo limitado

Los puzzles o juegos de lógica dentro del Juego Serio contienen un cronómetro el cual indica al jugador el plazo máximo que tiene para resolverlo, sin embargo, esto conlleva tiempo, tal es el caso de los puzzles de “Representación de imágenes” donde inicialmente se tenía que dibujar la matriz gráfica píxel por píxel.

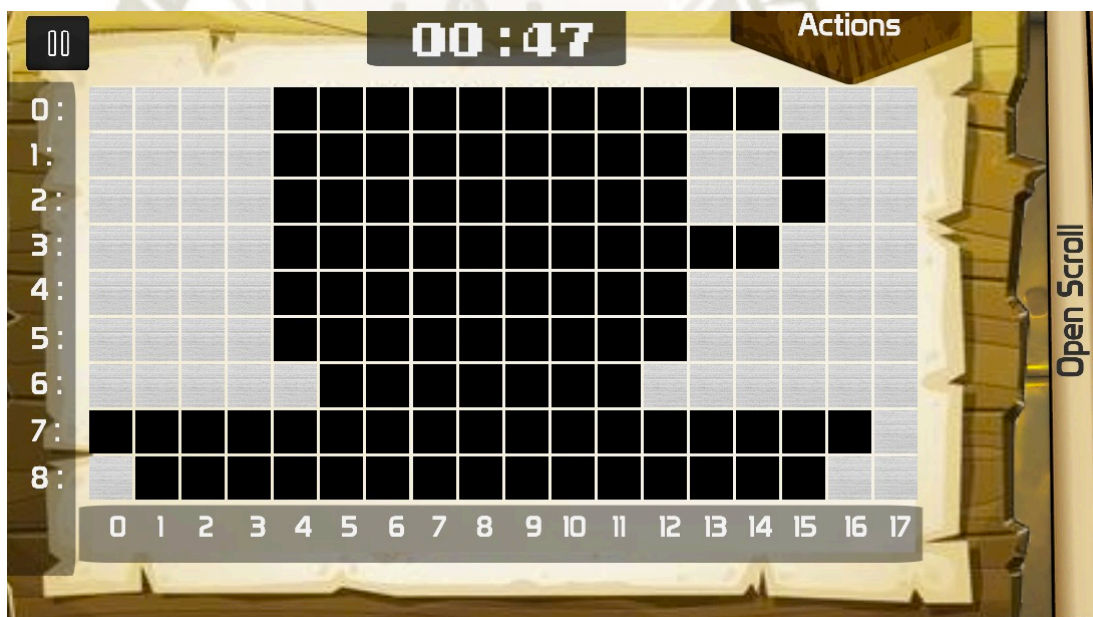


Figura 4.103 Actividad II del puzzle de Representación de imágenes

Fuente: Elaboración propia (2018)

Dada esta casuística presentada, se implementó un mecanismo que permite graficar a través del evento arrastrar y soltar del ratón, esto ayudó a completar el ejercicio de manera más eficiente.

b) Audios Descriptivos

Inicialmente se propuso que las pistas o “Hints” dentro del juego se presentan de manera audiovisual, esto quiere decir que el jugador podía escuchar el contenido de la pista mientras la observaba gráficamente, sin embargo, se estableció que una vez que el jugador iniciaba el audio este debía ser escuchado hasta el final, restringiendo así la posibilidad de detenerlo por decisión del usuario.



Figura 4.104 Vista audio-descriptiva de una Pista o Hint

Fuente: Elaboración propia (2018)

Este procedimiento, ocasionaba que el jugador se bloquee y pierda la concentración del juego de lógica. Acción que se pudo validar en la lectura desde el dispositivo Emotiv donde se evidenciaba que el nivel de estrés aumentaba en ese punto. Por lo tanto, dada esta problemática se introdujo una mejora, que consistía en darle la opción al jugador de detener el audio cuando este lo deseara así.

5.7. Pruebas Finales

En una segunda oportunidad incrementando las mejoras del punto anterior, se aplicó el juego a los grupos 4 y 6 de Pensamiento Computacional y 1 y 2 de Programación II del año 2017 y grupo 1 y 2 Pensamiento Computacional del año 2018, llevándose a cabo el mismo programa de ejecución de la instancia inicial (Grupo 3). Asimismo, se solicitó el respectivo permiso a los estudiantes que participaron del estudio para utilizar sus datos e imágenes para el proyecto de investigación.



Figura 4.105 Aplicación del Juego Serio en el grupo 4 de Pensamiento Computacional
Fuente: Elaboración propia (2018)



Figura 4.106 Aplicación del Juego Serio en el grupo 2 Programación II (2018)
Fuente: Elaboración propia (2018)

En los grupos 1 y 2 de Pensamiento computacional y grupo 4 de Programación II del año 2017 y grupo 4 de Pensamiento Computacional del año 2018, se aplicó la metodología tradicional. Esto quiere decir, que no se utilizó el Juego Serio para introducir el contenido académico, en su lugar se utilizaron diapositivas con la dirección y la autorización del Ing. Ángel Montesinos (Docente a cargo) y estudiantes de dichos cursos para el tratamiento de sus datos e imágenes para fines del presente estudio.

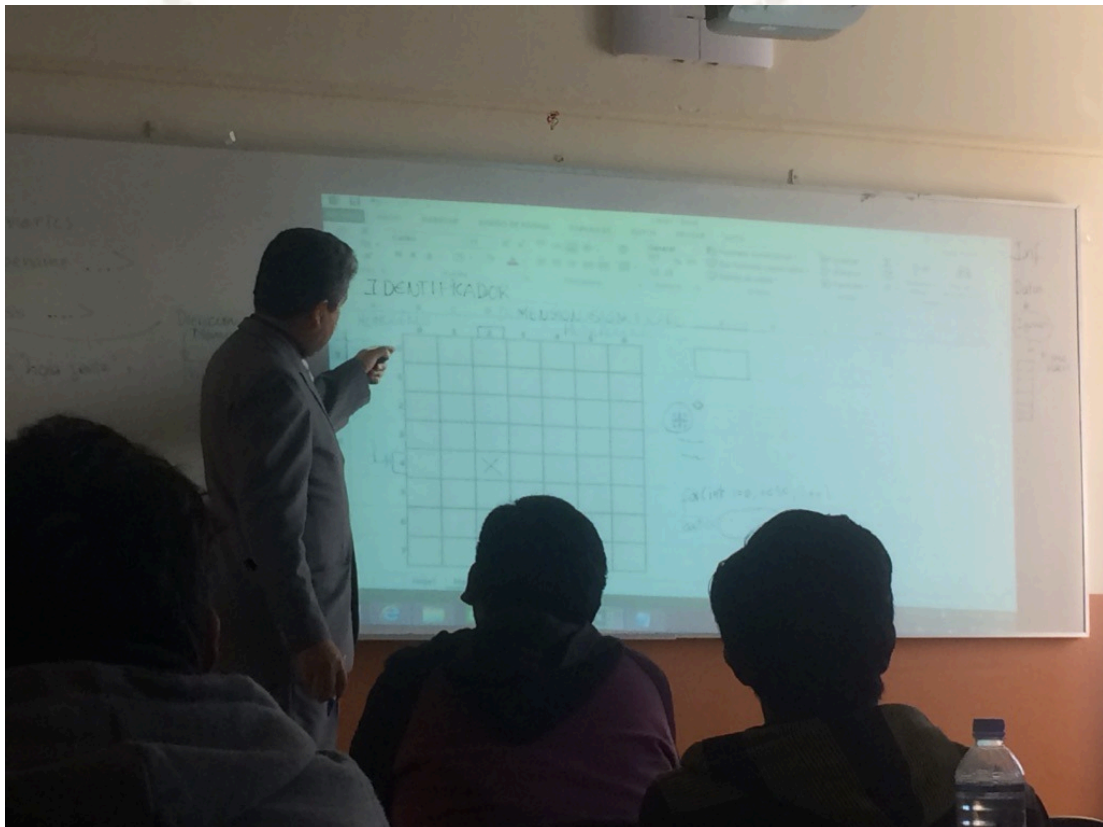
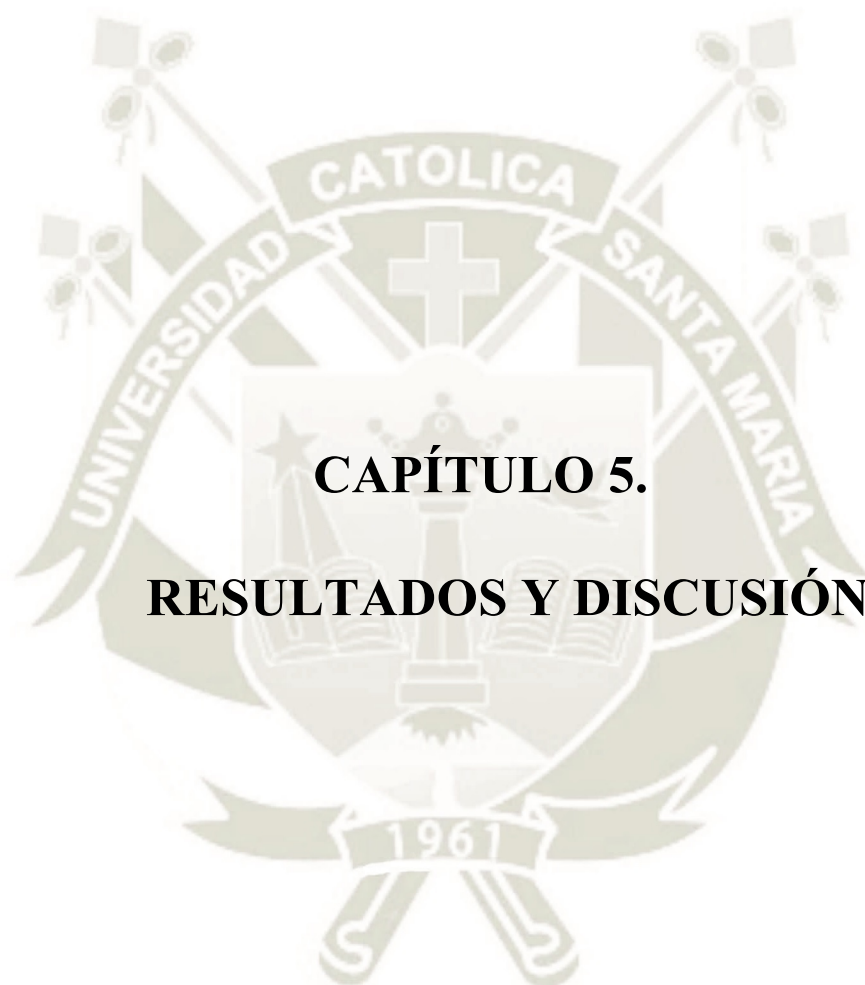


Figura 4.107 Clase presencial de matrices en el grupo 4 Programación II (2018)

Fuente: Elaboración propia (2018)

Al finalizar la actividad, se aplicó una encuesta de satisfacción (Ver anexo B) a un total de 24 de los estudiantes que utilizaron el Juego Serio como metodología de alternativa de aprendizaje. Obteniendo apreciaciones positivas sobre la propuesta, además de sugerencias de que mas cursos de programación deben enseñarse a través de Juegos Serios.



CAPÍTULO 5.

RESULTADOS Y DISCUSIÓN

1. Análisis de los resultados

En una primera instancia se debe verificar si la muestra que se ha obtenido presenta normalidad en los datos, para ello utilizaremos una prueba de normalidad como Kolmogorov-Smirnov, de ser así utilizaremos la prueba T-student, caso contrario aplicaremos la prueba de Wilcoxon de los rangos con signo para muestras no paramétricas. Esto nos permite determinar si existen diferencias entre las muestras. Posteriormente para reforzar las conclusiones que podemos obtener de los resultados, se debe verificar las correlaciones con otras variables como el ranking, el tiempo de juego y el número de niveles jugados. Para ello utilizaremos el método de Pearson para datos paramétricos y Spearman para datos no paramétricos.

Los resultados obtenidos indican que tanto la metodología tradicional como la alternativa generan el mismo grado de conocimiento en los estudiantes. Pero la alternativa tiene mayor impacto motivacional (datos obtenidos con el emotiv).

grupo		N	Media	Desviación estándar	Mínimo	Máximo
tradicional	pretest	23	2,8130	2,29907	1,00	10,20
	postest	23	13,1826	2,74850	8,00	19,00
videojuego	pretest	61	4,2131	2,14177	1,00	10,50
	postest	61	15,3541	2,86643	8,00	20,00
videjuego en parejas	pretest	13	8,8846	2,44228	6,00	13,50
	postest	13	17,9231	1,20149	15,00	20,00

Figura 5.1 Estadísticos descriptivos Generales

Fuente: Elaboración propia (2018)

Descriptive Statistics

grupo		N	Mean	Std. Deviation	Minimum	Maximum
videojuego	pretest	55	3.2636	1.99038	.00	10.50
	posttest	55	14.0418	2.22242	10.00	20.00
tradicional	pretest	25	2.6800	2.26219	.30	10.20
	posttest	25	12.5200	3.57036	2.30	19.00

Figura 5.2 Estadísticos descriptivos entre metodologías

Fuente: Elaboración propia (2018)

Ranks

	grupo	N	Mean Rank
diferencia	videojuego	55	42.50
	tradicional	25	36.10
	Total	80	

Figura 5.3 Rangos entre metodologías

Fuente: Elaboración propia (2018)

Test Statistics^{a,b}

	diferencia
Kruskal-Wallis H	1.307
df	1
Asymp. Sig.	.253

a. Kruskal Wallis Test

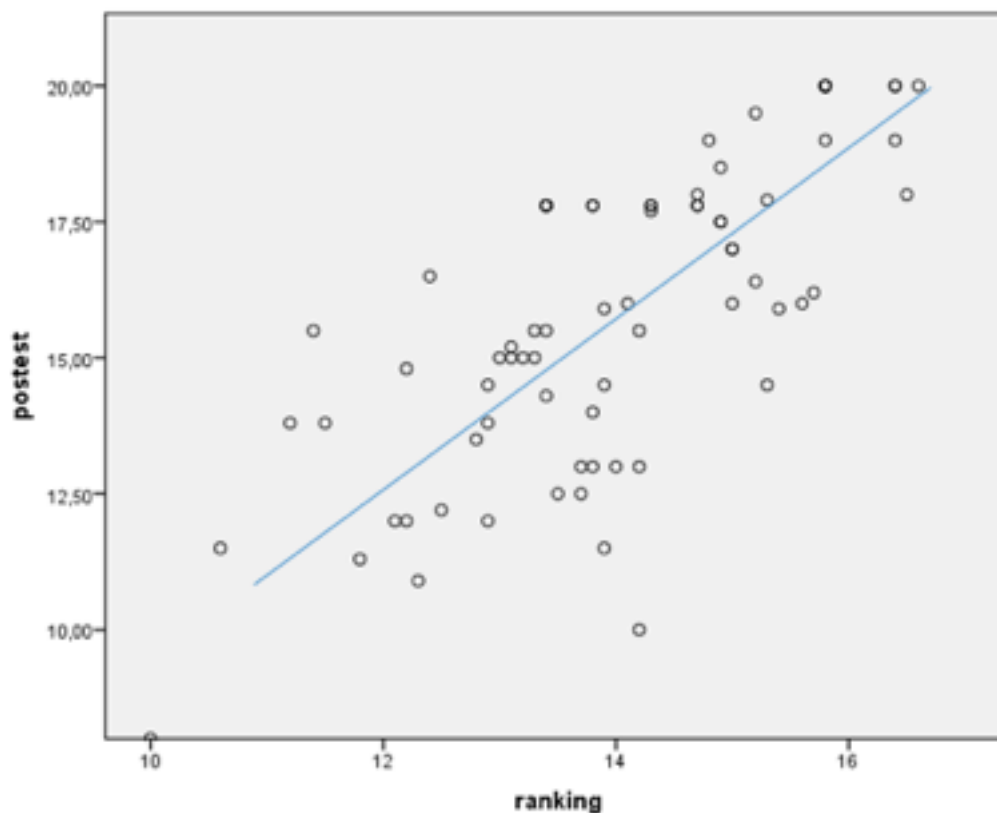
b. Grouping Variable:
grupo

Figura 5.4 Test Kruskal-Wallis: No diferencia significativa

Fuente: Elaboración propia (2018)

La prueba de Kruskal-Wallis da como resultado un valor $p=0.253$, por lo tanto al ser $p>0.05$ aceptamos la hipótesis nula (H_0 = El aprendizaje es significativo para ambos grupos).

Así mismo a través de la prueba de Spearman se ha probado la relación entre los resultados obtenidos en el post-test con el ranking obtenido (a mayor ranking en el juego mayor es el puntaje obtenido en el post-test)



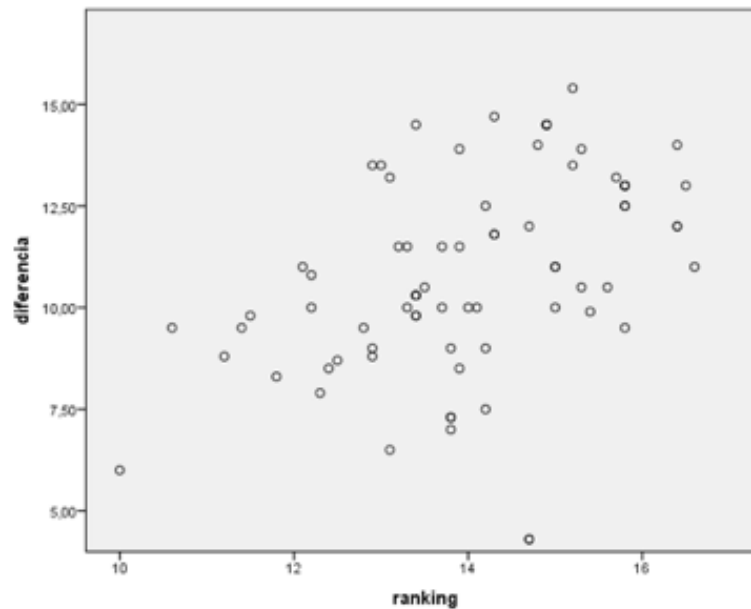
Rho de Spearman: 0,734

Figura 5.5 Prueba de correlación entre variables Spearman (post-test y ranking)

Fuente: Elaboración propia (2018)

La correlación está relacionada con la pendiente, extrapolando los puntos se puede graficar una la línea que viene a ser la pendiente que sería $r = 0.74$.

Utilizando la diferencia entre post-test y pre-test como una variable para demostrar que si al tener un mejor ranking explica que el aprendizaje obtenido es mayor en comparación con lo que se tenía al inicio (pre-test) la gráfica resultó ser más dispersa.



Rho de Spearman: 0,478

Figura 5.6 Prueba de correlación entre variables Spearman (diferencia y ranking)

Fuente: Elaboración propia (2018)

Lo cual tiene sentido ya que el ranking es una variable que no guarda alguna relación con el pre-test (variable necesaria para obtener la diferencia pre-test/post-test)

Estadísticas de grupo

	niveles	N	Media	Desviación estándar	Media de error estándar
postest	2	10	13,2700	1,99335	,63035
	3	62	16,1952	2,75654	,35008

P: 0,002

(T student)

Figura 5.7 Estadísticas de grupo (T-Student)

Fuente: Elaboración propia (2018)

El siguiente análisis con T-student con relación con el post-test y la cantidad de niveles jugados indica que efectivamente a mayor cantidad de niveles jugados es mayor el puntaje obtenido en el post-test.

Los resultados anteriores fueron respaldados por un segundo análisis sobre la correlación entre las variables de la muestra. Finalmente se analizó los datos obtenidos del dispositivo Emotiv como se puede observar en la Figura 5.8. Demostrando una diferencia en a nivel de compromiso donde el segundo cuartil Q2 (una mediana más alta) entre estas dos metodologías.

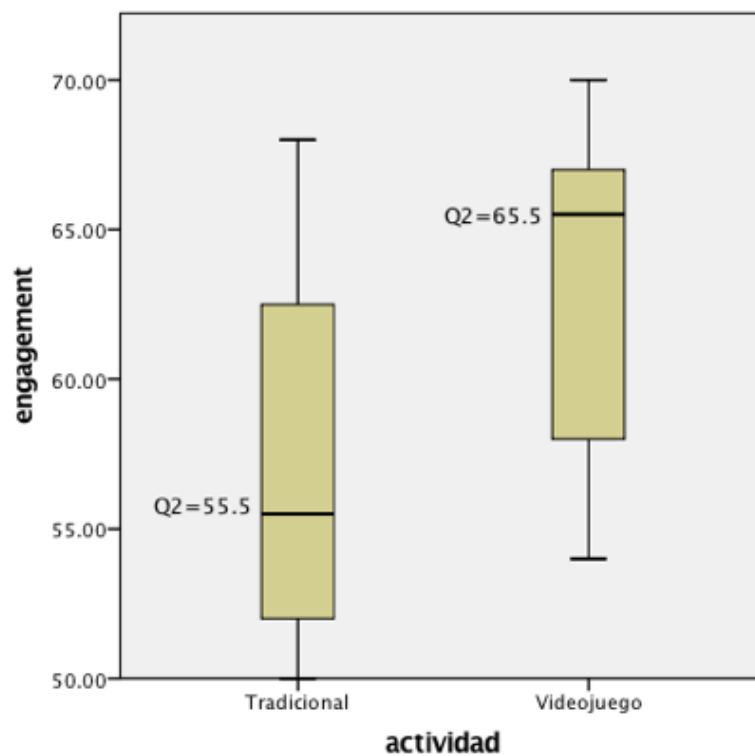


Figura 5.8 Diagrama de caja y bigotes (Nivel de compromiso/Metodología)

Fuente: Elaboración propia (2018)

Statistics			Statistics		
engagement			engagement		
N	Valid	4	N	Valid	6
	Missing	0		Missing	0
Minimum		50.00	Minimum		54.00
Maximum		68.00	Maximum		70.00
Percentiles	25	51.0000	Percentiles	25	57.0000
	50	55.5000		50	65.5000
	75	65.2500		75	67.7500

Figura 5.9 Estadísticos descriptivos de percentiles (Caja y bigotes)

Fuente: Elaboración propia (2018)

2. Discusión de los resultados

Esta sección relaciona los resultados obtenidos con la literatura recabada, el estado de la cuestión y la propia investigación con la finalidad de validar justificadamente el aporte del presente trabajo.

Para verificar que los resultados sean comparados con en el estado del arte, se detalla a continuación, un contraste de los resultados con los trabajos previos del área de la investigación.

De acuerdo con los estadísticos descriptivos presentados en la sección de resultados, quedó demostrado que tanto el Juego Serio como la metodología tradicional tuvieron el mismo impacto en cuestión de conocimientos, lo cual se alinea con los resultados obtenidos en los estudios de [31]–[35] sobre la efectividad de Juegos Serios en educación (Edutainment) y específicamente en la formación en la carrera de Ingeniería de Sistemas y tecnología [35], [38]–[40].

Respecto a la adaptación del enfoque Test Driven Learning (TDL) en la fase de diseño de casos de prueba, los estudiantes de los cursos de Pensamiento computacional y Programación

II, lograron superar los ejercicios (Suma y multiplicación de matrices) sin dificultad, lo cual se pudo evidenciar en las puntuaciones obtenidas al final del juego, posteriormente respaldados por el post-test. Contra argumentando los estudios de [15], [18] los cuales indicaron que los estudiantes en esta etapa suelen presentar una resistencia al momento de escribir pruebas. Podemos decir que la integración del TDL y un Juego Serio mejora la adopción del enfoque en los estudiantes principiantes, dado que estos se presentan bajo una perspectiva más afable. Por otro lado, no se pudo validar si los estudiantes que realizaban una caso de prueba (Test-first) escribían más casos de prueba que los que utilizaban el Test-last descrito en [14], [25] dado que el juego se diseñó únicamente para realizar pruebas (Test-first).

Respecto a los obstáculos expuestos en [9], [19] acerca de la implementación de Test Driven Learning en el plan de estudios de la carreras de informática. De acuerdo a los resultados evidenciados, se puede decir que al generalizar o encapsular los conceptos sobre los conceptos de Testing y casos de prueba, que según [15] son fundamentales en el proceso TDL, se pueden superar transformando los conceptos sobre Testing en objetivos del juego, omitiendo así la ruta de aprendizaje (POO → Testing → Contenido académico) inicialmente propuesto por [15] que ocasiona un desenfoque en el objetivo de estudio (Contenido académico) [25] y por otro lado favorecía la resistencia en estudiantes ya mencionada, debido a la carencia de conocimientos y falta de experiencia del estudiante.

Entonces, para garantizar la calidad del método propuesto (TDL + Juego Serio) y determinar las limitaciones o alcances, es necesario realizar un análisis reflexivo con la finalidad de mejorar la calidad de la línea de investigación en cuestión.

En primer lugar, una de las falencias encontradas en el método propuesto fue el diseño implementado bajo una estructura rígida. Esto quiere decir, que el juego no permite adaptarse sobre otros conceptos que no sean Matrices y Transformaciones geométricas. Por naturaleza el diseño de casos de prueba debe ser un procedimiento creativo para el desarrollador, y nuestro

método presenta limitaciones en este aspecto, siendo estas solo de uso exclusivo para evaluar algoritmos con matrices (Suma y Multiplicación de matrices). Esto conlleva a generar una dependencia del lenguaje y la estructura [14]. Sin embargo, la implementación del componente (LevelGenerator) para generar niveles aleatorios abre la posibilidad de crear también puzzles aleatorios con una distinta temática y dificultad subsanando así la limitante de una estructura rígida.

En segundo lugar, el uso de un mecanismo automatizado para la evaluación de los estudiantes permite generar las calificaciones de manera más rápida y facilitar así la entrega de retroalimentación a los estudiantes al momento de resolver los ejercicios durante el proceso TDL. Validando así, el aporte de [26] y recomendaciones de D. Janzen y Saiedian (2008) para facilitar la implementación de TDL en cursos de primeros años. Sin embargo, dado que la retroalimentación fue pre-diseñada e introducida al juego de acuerdo con las acciones del estudiante, la metodología propuesta presenta una cobertura limitada al momento devolver retroalimentación específica.

Al inicio de este estudio se plantearon tres preguntas de investigación, las cuales demarcan la hipótesis de la investigación, a continuación, se entregan las posibles razones por las cuales los resultados obtenidos satisfacen o no estos cuestionamientos iniciales.

En relación con la primera pregunta de investigación: ¿La aplicación de juegos serios facilita la adopción del enfoque TDL en los estudiantes? De acuerdo con los resultados obtenidos se puede decir que un Juego Serio encapsula las limitantes conceptuales sobre Testing en el proceso TDL, y facilita su adopción centrando su objetivo al contenido académico (para nuestro caso de estudio: Matrices y transformaciones geométricas).

Continuando con la segunda pregunta de investigación: ¿Cómo se puede implementar Test Driven Learning dentro de un Juego Serio? El desarrollo de un Juego Serio que implemente

TDL se determinó en base en la construcción de 5 artefactos: a) Diseño de interfaces y de software, b) Una estructura que soporte una evaluación de algoritmos, c) Un entorno para la creación de casos de prueba, d) Un IDE para programación con bloques y e) Un sistema de calificación automatizado. De esta manera, se puede llevar a cabo un proceso TDL inmerso en un Juego Serio, donde los estudiantes puedan escribir casos de prueba de forma interactiva [20].

Respecto a la tercera y última pregunta: ¿Es posible reducir el tiempo del aprendizaje en programación utilizando TDL en un Juego Serio? Según nuestros registros obtenidos, si bien se logró aplicar el Juego Serio en un lapso de 02 horas sin complicaciones, lo fue también para la clase tradicional. Al analizar directamente los datos se puede observar que en el tiempo de juego en promedio es 30 minutos, sin embargo, este valor guarda similitud con el tiempo presupuestado en el plan de ejecución visto previamente antes de la aplicación del juego. Los estudiantes al tener un límite de tiempo para interactuar con el juego, ocasiona que algunos de ellos no completen todos los niveles del juego. Por otro lado, también se pudo observar que aquellos que disponían de tiempo, ejecutaban nuevamente el juego con la finalidad de obtener mejores puntajes. Por lo tanto, dadas estas características no se puede asegurar que exista una diferencia significativa en el tiempo de ejecución entre ambos métodos (Juego Serio y Metodología Tradicional).

CONCLUSIONES

PRIMERO. Se desarrolló con éxito un Juego Serio que integra el enfoque Test Driven Learning utilizando UNITY en base a una metodología de desarrollo ágil (SCRUM) como método alternativo para la enseñanza de Matrices y Aplicaciones en Transformaciones geométricas. Asimismo, se documentó cada una de las etapas de su construcción, lo que facilitará la elaboración de productos similares en posteriores investigaciones.

SEGUNDO. Se determinó estadísticamente que tanto la metodología tradicional como la metodología alternativa (TDL + Juego Serio) fueron similarmente efectivas en cuanto a conocimientos.

TERCERO. Quedó demostrado del análisis de los datos neuro-sensoriales obtenidos (estrés, interés, compromiso, emoción, concentración y relajación) por el dispositivo Emotiv, que la metodología propuesta genera un aporte cognitivo mayor en cuanto al compromiso de los estudiantes al utilizar videojuegos como herramienta de aprendizaje.

CUARTO. Se verificó que en cuantos más niveles de juego son jugados por los estudiantes, mayores son los puntajes obtenidos debido a que se alcanzaba una mayor cobertura de los contenidos. Asimismo, aquellos estudiantes que realizaron el juego por segunda o tercera vez mostraban una mejora en su puntaje respecto a los que solo habían jugado una vez.

QUINTO. El utilizar un mecanismo de evaluación automatizada sobre el enfoque de enseñanza basada en pruebas (TDL) redujo considerablemente el tiempo de calificación y la entrega de retroalimentación a los estudiantes.

SEXTO. Los estudiantes que utilizaron el Juego Serio como método de aprendizaje, demostraron una apreciación positiva al uso de videojuegos como alternativa en la enseñanza de programación lo cual quedó demostrado en a través del cuestionario de satisfacción.

TRABAJOS FUTUROS

Durante la realización del estudio se precisaron ciertas limitaciones que impiden generalizar los resultados para futuras investigaciones las cuales fueron expuestas la discusión de los resultados, en esta sección se plantearán proyecciones que permitan fortalecer y dar a pie a nuevos estudios.

Como se pudo evidenciar la entrega de retroalimentación del Juego Serio agrupaba solo las casuísticas mas comunes, por lo tanto, se plantea para futuras investigaciones adicionar un sistema autónomo de datos que permita incrementar el alcance horizontal y vertical (calidad y cantidad) en la retroalimentación del Juego Serio.

Dado que el Juego Serio se ha estructurado bajo una temática específica (Matrices y Transformaciones geométrica) se enfatiza para una continuación del método en la construcción de un generador para juegos de lógica (Puzles) con temática programable, similar a la de generación de niveles.

El enfoque Test Driven Learning ha demostrado ser eficaz en la enseñanza de programación y a través de los Juegos Serios se han minimizado las limitaciones en su adopción en una etapa primaria. Sin embargo, el costo en desarrollo de un Juego Serio se presenta como un nuevo obstáculo que dificulta su aplicación como método de enseñanza. Por lo tanto, surge la necesidad de realizar nuevos estudios en la implementación TDL con Juegos Serios con el objetivo de reducir el tiempo en el desarrollo.

Finalmente, se proyecta realizar un nuevo estudio utilizando el dispositivo Emotiv EPOC de 14 canales en lugar del dispositivo utilizado en el presente trabajo (Emotiv Insight de 5 canales) con la finalidad de obtener datos más específicos y un análisis más completo sobre la eficacia del Juego Serio.

REFERENCIAS

- [1] D. D. Budny and C. A. Delaney, “Working with students and parents to improve the freshman retention,” *31st ASEE/IEEE Front. Educ. Conf.*, no. December, p. 6, 2001, doi: 10.1109/FIE.2001.963918.
- [2] C. Baillie and G. Fitzgerald, “Motivation and attrition in engineering students,” *Eur. J. Eng. Educ.*, vol. 25, no. 2, pp. 145–155, 2000, doi: 10.1080/030437900308544.
- [3] Z.-H. Chen, T.-C. Chien, and T.-W. Chan, “Using Self-competition to Enhance Students’ Learning,” 2011, pp. 234–235.
- [4] M. Besterfield-Sacre, C. J. Atman, and L. J. Shuman, “Characteristics of Freshman Engineering Students: Models for Determining Student Attrition in Engineering,” *J. Eng. Educ.*, vol. 86, no. 2, pp. 139–149, 1997, doi: 10.1002/j.2168-9830.1997.tb00277.x.
- [5] C. Moller-Wong, Arvid, and Eide, “An engineering student retention study,” *J. Eng. Educ.*, vol. 86, no. 1, pp. 7–15, 1997, doi: 10.1002/j.2168-9830.1997.tb00259.x.
- [6] M. del P. Mori Sánchez, “DESERCIÓN UNIVERSITARIA EN ESTUDIANTES DE UNA UNIVERSIDAD PRIVADA DE QUITOS,” *Rev. Digit. Investig. en Docencia Univ.*, vol. 6, no. 1, p. 60, Dec. 2012, doi: 10.19083/ridu.6.42.
- [7] Oficina de Informática Universidad Católica de Santa María, “Datos estadísticos de deserción estudiantil en la Escuela Profesional de Ingeniería de Sistemas.” UCSM, Arequipa, 2015.
- [8] M. E. Echeveste and M. C. Martínez, “Desafíos en la enseñanza de Ciencias de la Computación Challenges of teaching Computer Science,” *Virtualidad, Educ. y Cienc.*, vol. 7, no. 12, pp. 34–48, 2016, [Online]. Available: <https://dialnet.unirioja.es/servlet/articulo?codigo=5591257>.

- [9] J. Clements and D. Janzen, “Overcoming Obstacles to Test-Driven Learning on Day One,” in *2010 Third International Conference on Software Testing, Verification, and Validation Workshops*, Apr. 2010, pp. 448–453, doi: 10.1109/ICSTW.2010.33.
- [10] J. L. G. Sánchez, F. L. G. Vela, F. M. Simarro, and N. Padilla-Zea, “Playability: analysing user experience in video games,” *Behav. Inf. Technol.*, vol. 31, no. 10, pp. 1033–1054, 2012, doi: 10.1080/0144929X.2012.710648.
- [11] L. Shuman, C. Delaney, H. Wolfe, A. Scalise, and M. Besterfield-Sacre, “Engineering attrition: Student characteristics and educational initiatives,” *Proc. Am. Soc. Eng. Educ.*, 1999.
- [12] T. Dvornik, D. S. Janzen, J. Clements, and O. Dekhtyar, “Supporting introductory test-driven labs with WebIDE,” in *2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEET)*, May 2011, pp. 51–60, doi: 10.1109/CSEET.2011.5876137.
- [13] D. S. Janzen, J. Clements, and M. Hilton, “An evaluation of interactive test-driven labs with WebIDE in CS0,” in *2013 35th International Conference on Software Engineering (ICSE)*, May 2013, pp. 1090–1098, doi: 10.1109/ICSE.2013.6606659.
- [14] D. Janzen and H. Saiedian, “Test-driven learning in early programming courses,” in *Proceedings of the 39th SIGCSE technical symposium on Computer science education - SIGCSE '08*, 2008, p. 532, doi: 10.1145/1352135.1352315.
- [15] D. S. Janzen, H. Saiedian, and S. H. Janzen D.S., “Test-driven learning: Intrinsic integration of testing into the CS/SE curriculum,” *Proc. Thirty-Seventh SIGCSE Tech. Symp. Comput. Sci. Educ.*, pp. 254–258, 2007, doi: 10.1145/1121341.1121419.
- [16] D. S. Janzen and J. Ryoo, “Engaging the net generation with evidence-based software engineering through a community-driven web database,” *J. Syst. Softw.*, vol. 82, no. 4,

- pp. 563–570, 2009, doi: 10.1016/j.jss.2008.12.047.
- [17] S. M. Embury, M. Borizanov, and C. Jay, “Red-Green-Go! A Self-Organising Game for Teaching Test-Driven Development,” in *Agile and Lean Concepts for Teaching and Learning*, Singapore: Springer Singapore, 2019, pp. 415–441.
- [18] T. Briggs and C. Dudley Girard, “Tools and techniques for test-driven learning in CS1,” in *Journal of Computing Sciences in Colleges*, 2007, pp. 37–43, [Online]. Available: <http://dl.acm.org/citation.cfm?id=1181854>.
- [19] L. P. Scatalon, E. F. Barbosa, and R. E. Garcia, “Challenges to integrate software testing into introductory programming courses,” *Proc. - Front. Educ. Conf. FIE*, vol. 2017-
Octob, pp. 1–9, 2017, doi: 10.1109/FIE.2017.8190557.
- [20] R. Stejskal and H. Siy, “Test-driven learning in high school computer science,” in *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)*, May 2013, pp. 289–293, doi: 10.1109/CSEET.2013.6595263.
- [21] CS Education Research Group, “Image Representation,” 2014. <http://csunplugged.org/image-representation/>.
- [22] F. Liarokapis, K. Debattista, A. Vourvopoulos, P. Petridis, and A. Ene, “Comparing interaction techniques for serious games through brain-computer interfaces: A user perception evaluation study,” *Entertain. Comput.*, vol. 5, no. 4, pp. 391–399, 2014, doi: 10.1016/j.entcom.2014.10.004.
- [23] M. L. Wu and K. Richards, “Facilitating Computational Thinking through Game Design,” 2011, pp. 220–227.
- [24] M. Haungs, C. Clark, J. Clements, and D. Janzen, “Improving first-year success and retention through interest-based CS0 courses,” *Proc. 43rd ACM Tech. Symp. Comput. Sci. Educ. - SIGCSE '12*, p. 589, 2012, doi: 10.1145/2157136.2157307.

- [25] C. Desai, D. Janzen, and K. Savage, “A survey of evidence for test-driven development in academia,” *ACM SIGCSE Bull.*, vol. 40, no. 2, pp. 97–101, Jun. 2008, doi: 10.1145/1383602.1383644.
- [26] D. Truscan, T. Ahmad, and C. H. Tran, “Applying Test-Driven Development for Improved Feedback and Automation of Grading in Academic Courses on Software Development,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12271 LNCS, no. September, 2020, pp. 310–323.
- [27] L. P. Scatalon, R. E. Garcia, and E. F. Barbosa, “Teaching Practices of Software Testing in Programming Education,” *Proc. - Front. Educ. Conf. FIE*, vol. 2020-Octob, 2020, doi: 10.1109/FIE44824.2020.9274256.
- [28] M. Hilton and D. S. Janzen, “On Teaching Arrays with Test-Driven Learning in WebIDE,” *ITiCSE*, pp. 93–98, 2012, doi: 10.1145/2325296.2325322.
- [29] C.-Y. Yang, L.-R. Chien, D. J. Buehrer, and C.-M. Chen, “An Evaluation on Interaction between APAS, TDD and Learning Style,” in *2009 International Conference on New Trends in Information and Service Science*, Jun. 2009, pp. 350–355, doi: 10.1109/NISS.2009.275.
- [30] E. J. Pozsgai Hernani, “Diseño de tareas que contribuyan a un aprendizaje significativo del concepto de derivada en estudiantes de ciencias administrativas,” Pontificia Universidad Católica del Perú, 2014.
- [31] S. Ramirez-Rosales, S. Vazquez-Reyes, J. L. Villa-Cisneros, and M. De Leon-Sigg, “A Serious Game to Promote Object Oriented Programming and Software Engineering Basic Concepts Learning,” in *2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT)*, Apr. 2016, pp. 97–103, doi:

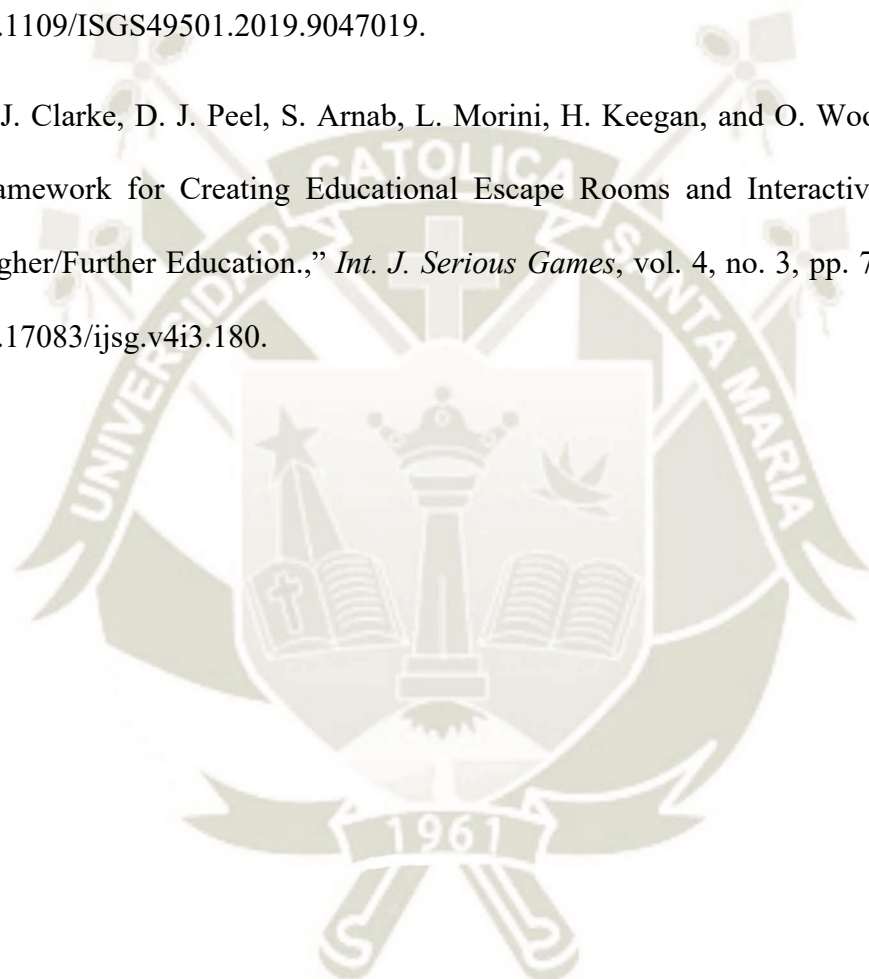
10.1109/CONISOFT.2016.23.

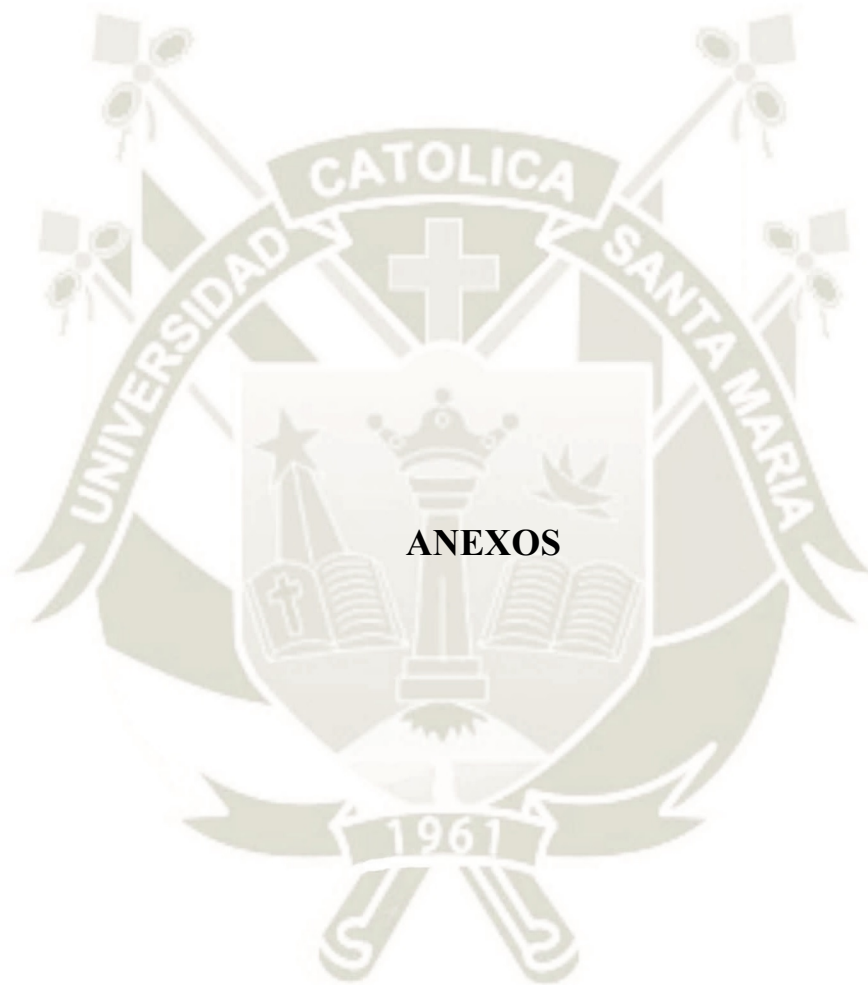
- [32] M. C. Gomez-Alvarez, G. P. Gasca-Hurtado, and C. F. Garcia Giraldo, “Juegos serios como estrategia de enseñanza del proceso de gestión de riesgos de software: Prototipo de un videojuego Serious games as a teaching strategy for software projects risk management videogame prototype,” in *2016 IEEE 11th Colombian Computing Conference (CCC)*, Sep. 2016, pp. 1–8, doi: 10.1109/ColumbianCC.2016.7750804.
- [33] P. Wouters, *Instructional Techniques to Facilitate Learning and Motivation of Serious Games*. Cham: Springer International Publishing, 2017.
- [34] C. S. Longstreet and K. Cooper, “A Meta-model for Developing Simulation Games in Higher Education and Professional Development Training,” *17th Int. Conf. Comput. Games*, pp. 39–44, 2012, doi: 10.1109/CGames.2012.6314549.
- [35] S. Y. bin S. Hussain, T. W. Hoe, and M. Z. bin Idris, “Digital game based learning: A new method in teaching and learning mathematics,” 2017, vol. 030016, p. 030016, doi: 10.1063/1.4983894.
- [36] M. J. Callaghan, N. McShane, and A. G. Eguiluz, “Using game analytics to measure student engagement/retention for engineering education,” in *Proceedings of 2014 11th International Conference on Remote Engineering and Virtual Instrumentation, REV 2014*, 2014, doi: 10.1109/REV.2014.6784174.
- [37] H. Duin, B. Pourabdollahian, K. D. Thoben, and M. Taisch, “On the effectiveness of teaching sustainable global manufacturing with serious gaming,” *2013 Int. Conf. Eng. Technol. Innov. ICE 2013 IEEE Int. Technol. Manag. Conf. ITMC 2013*, 2015, doi: 10.1109/ITMC.2013.7352654.
- [38] S. Y. S. Hussain, W. H. Tan, and M. Z. Idris, “Digital Game-Based Learning for Remedial Mathematics Students: A New Teaching and Learning Approach in Malaysia,”

- Int. J. Multimed. Ubiquitous Eng.*, vol. 9, no. 11, pp. 325–338, Nov. 2014, doi: 10.14257/ijmue.2014.9.11.32.
- [39] V. Aleven, E. Myers, M. Easterday, and A. Ogan, “Toward a framework for the analysis and design of educational games,” in *DIGITEL 2010 - The 3rd IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, 2010, doi: 10.1109/DIGITEL.2010.55.
- [40] D. Carrington, A. Baker, and A. van der Hoek, “All in the Game: Teaching Software Process Concepts,” in *Proceedings Frontiers in Education 35th Annual Conference*, 2004, pp. F4G-13-F4G-18, doi: 10.1109/FIE.2005.1612152.
- [41] V. Riemer and C. Schrader, “Learning with quizzes, simulations, and adventures: Students’ attitudes, perceptions and intentions to learn with different types of serious games,” *Comput. Educ.*, 2015, doi: 10.1016/j.compedu.2015.05.003.
- [42] B. Kenwright, “Brief review of video games in learning & education how far we have come,” in *SIGGRAPH Asia 2017 Symposium on Education*, Nov. 2017, pp. 1–10, doi: 10.1145/3134368.3139220.
- [43] H. Mahmoudi, M. Koushfar, J. A. Saribagloo, and G. Pashavi, “The effect of computer games on speed, attention and consistency of learning mathematics among students,” *Procedia -Social Behav. Sci.*, vol. 176, pp. 419–424, 2015, doi: 10.1016/j.sbspro.2015.01.491.
- [44] R. Hunicke, M. Leblanc, and R. Zubek, “MDA: A formal approach to game design and game research,” *AAAI Work. - Tech. Rep.*, vol. WS-04-04, pp. 1–5, 2004.
- [45] P. Sancho, R. Fuentes-Fernández, and B. Fernández-Manjón, “NUCLEO: Adaptive Computer Supported Collaborative Learning in a role game based scenario,” in *Proceedings - The 8th IEEE International Conference on Advanced Learning*

- Technologies, ICALT 2008*, 2008, doi: 10.1109/ICALT.2008.147.
- [46] A. Adorjan and I. F. de Kereki, “Design of activities for CS1: A competences oriented approach (unpacking the Informed Design Teaching and Learning Matrix),” in *2013 XXXIX Latin American Computing Conference (CLEI)*, Oct. 2013, pp. 1–6, doi: 10.1109/CLEI.2013.6670622.
- [47] R. Orji, R. L. Mandryk, and J. Vassileva, “Improving the Efficacy of Games for Change Using Personalization Models,” *ACM Trans. Comput. Interact.*, vol. 24, no. 5, pp. 1–22, 2017, doi: 10.1145/3119929.
- [48] F. A. G. Hernández, F. L. Madriz, and C. M. C. Esquivel, “Metodologías para el Desarrollo de Videojuegos Serios: Una Revisión de Literatura,” *Tecnol. Educ. Rev. CONAIC*, vol. 6, no. 1, pp. 103–114, 2019, doi: 10.32671/terc.v6i1.85.
- [49] K. Schwaber, *SCRUM Development Process*, no. December. 2020.
- [50] A. Vourvopoulos and F. Liarokapis, “Evaluation of commercial brain–computer interfaces in real and virtual world environment: A pilot study,” *Comput. Electr. Eng.*, vol. 40, no. 2, pp. 714–729, 2014, doi: 10.1016/j.compeleceng.2013.10.009.
- [51] J. Stewart, *Cálculo de una Variable*. 2012.
- [52] Bjarne Stroustrup, *The C++ Programming Language*. 2013.
- [53] S. I. Groosman, *Algebra Lineal*, 6th ed. 2012.
- [54] S. B. Melo, “TRANSFORMACIONES GEOMÉTRICAS SOBRE IMÁGENES DIGITALES,” pp. 1–13.
- [55] A. T. Spinelli and S. M. Massa, “Elicitación de Requerimientos Educativos en un Serious Game,” *XIII Congr. Tecnol. en Educ. y Educ. en Tecnol.*, no. July, pp. 68–76, 2018, [Online]. Available: <http://sedici.unlp.edu.ar/handle/10915/68892>.

- [56] P. Kruchten, “Planos Arquitectónicos: El Modelo de 4+ 1 Vistas de la Arquitectura del Software.,” *IEEE Softw.*, vol. 12, no. 6, pp. 42–50, 1995, [Online]. Available: http://alfredo.chacharaselnido.com/Desarrollo_proyectos/unidad1/4+1%5B1%5D.pdf.
- [57] F. F. Bakhsheshi, “Serious Games and Serious Gaming in Escape Rooms,” *Proc. 2019 Int. Serious Games Symp. ISGS 2019*, pp. 42–47, 2019, doi: 10.1109/ISGS49501.2019.9047019.
- [58] S. J. Clarke, D. J. Peel, S. Arnab, L. Morini, H. Keegan, and O. Wood, “EscapED: A Framework for Creating Educational Escape Rooms and Interactive Games to For Higher/Further Education.,” *Int. J. Serious Games*, vol. 4, no. 3, pp. 73–86, 2017, doi: 10.17083/ijsg.v4i3.180.





ANEXO A. Pre-test y Post-test



Universidad Católica de Santa María
Escuela Profesional de Ingeniería de Sistemas

Representación Imágenes y Matrices

Apellidos y nombres : _____

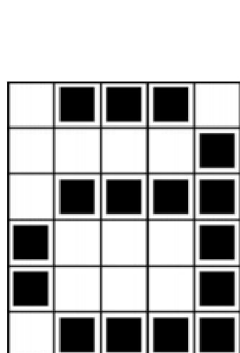
Código: _____

Fecha : _____

Indicaciones:

No está permitido utilizar apuntes ni material de apoyo, deberá utilizar solo lápiz  para su solución.

1. Si tuvieras la imagen de una letra “a”, graficada de la siguiente manera, explica el significado de los números que están del lado derecho:

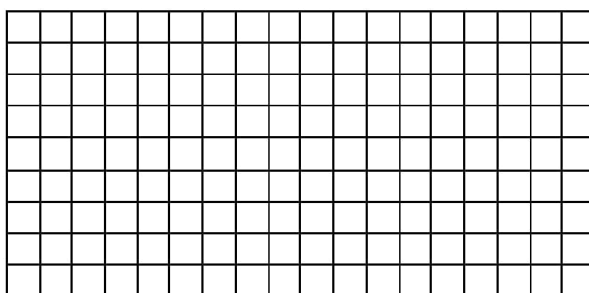


	■	■	■		
					■
	■	■	■	■	
■					■
■					■
	■	■	■	■	

1, 3, 1
4, 1
1, 4
0, 1, 3, 1
0, 1, 3, 1
1, 4

Respuesta :

2. Gráfica la siguiente imagen y responde la siguiente pregunta :



4, 11
4, 9, 2, 1
4, 9, 2, 1
4, 11
4, 9
4, 9
5, 7
0, 17
1, 15

3.

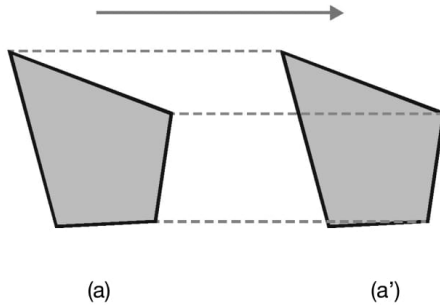
¿Cual es la imagen secreta?

3.



Universidad Católica de Santa María
Escuela Profesional de Ingeniería de Sistemas

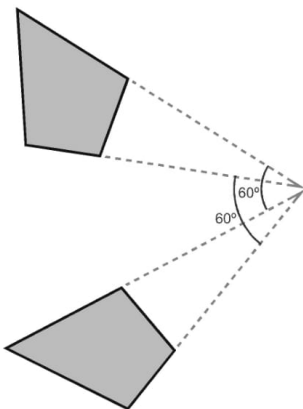
3. En un plano cartesiano. Cual seria la mejor forma de mover el polígono como se ilustra en la imagen?



Marca la opción correcta:

- a) Aplicar una fuerza en el eje x.
- b) Usar una traslación o transformación.
- c) Volver a graficar la imagen.
- d) Duplicar la imagen ('a) y eliminar (a).

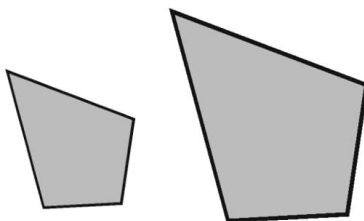
4. En un plano cartesiano. Cual seria la mejor forma de girar el polígono como se ilustra en la imagen?



Marca la opción correcta:

- e) Usar una rotación geométrica.
- f) Usar una traslación geométrica.
- g) Usar una inclinación geométrica.
- h) Modificamos el ángulo de inclinación.

5. En un plano cartesiano. Cual seria la mejor forma de aumentar el tamaño el polígono como se ilustra en la imagen?



Marca la opción correcta:

- i) Usar una interpolación geométrica.
- j) Usar un escalamiento geométrico.
- k) Usar una inclinación geométrica.
- l) Modificamos el ángulo de inclinación.



Universidad Católica de Santa María
Escuela Profesional de Ingeniería de Sistemas

⚠ OBSERVACIÓN

¿Que es una Matriz ?:

Es un conjunto de elementos ubicados en filas y columnas.

Una matriz en programación es similar a las cuadrículas que utilizamos en las preguntas anteriores.

6. ¿Cuando se dice que una **matriz es cuadrada**? Explica con un ejemplo.

7. ¿Cuándo se dice que dos **matrices son iguales**?

- a) Cuando tienen el mismo orden y los elementos son diferentes en cada posición.
- b) Cuando tienen diferente orden o dimensión y los elementos son iguales en cada posición.
- c) Cuando tienen los mismos elementos en las mismas posiciones.
- d) Cuando tienen la misma dimensión y los elementos que ocupan el mismo lugar en ambas son iguales.

8. ¿En una matriz de orden **p x q**, qué letra representa las **filas** y cuál las **columnas**?

- a) p las columnas y q las filas.
- b) p las filas y q las columnas.
- c) p x q el orden de la matriz.
- d) p y q deben ser iguales.

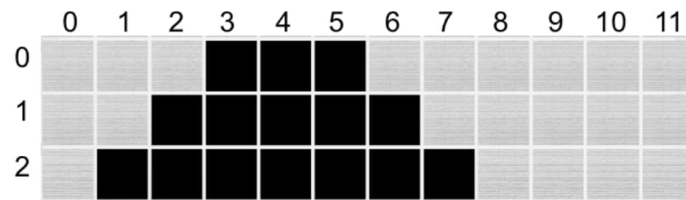
9. ¿A que se llama una **matriz transpuesta**?

- a) Matriz que se obtiene cambiando ordenadamente las filas por las columnas.
- b) Cuando una matriz está una sobre otra.
- c) Una matriz se se mueve una cierta cantidad de posiciones.
- d) Cuando se intercambian los elementos de una matriz

10. En la siguiente imagen **M** (Representada en pixeles ej. Pixel(0,3) = Negro)



Universidad Católica de Santa María
Escuela Profesional de Ingeniería de Sistemas



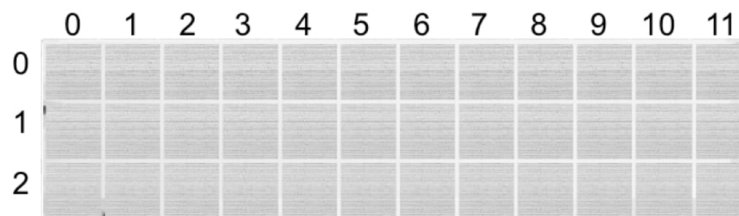
Representada con la siguiente matriz : (3 blancos, 3 negros, 6 blancos) para la primera fila

$$M = \begin{bmatrix} 3 & 3 & 6 \\ 2 & 5 & 5 \\ 1 & 7 & 4 \end{bmatrix}$$

a) Aplicar una traslación (+3 en el eje X) a la matriz **M**

$$M' = \begin{bmatrix} 3 & 3 & 6 \\ 2 & 5 & 5 \\ 1 & 7 & 4 \end{bmatrix} + \begin{bmatrix} \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \end{bmatrix} = \begin{bmatrix} \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \end{bmatrix}$$

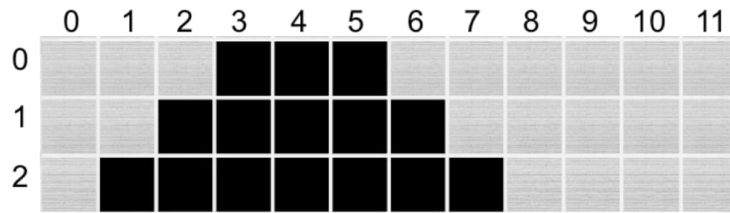
b) Graficar la imagen resultante (**M'**)



11. En la siguiente imagen **M** (Representada en pixeles e.g. P(0,3) = Negro)



Universidad Católica de Santa María
Escuela Profesional de Ingeniería de Sistemas



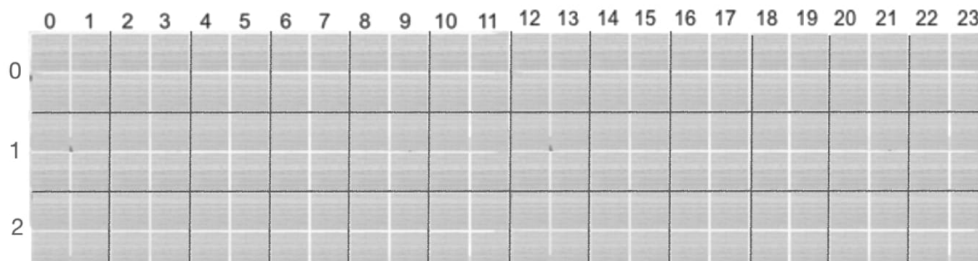
Representada con la siguiente matriz : (3 blancos, 3 negros, 6 blancos) para la primera fila

$$M = \begin{bmatrix} 3 & 3 & 6 \\ 2 & 5 & 5 \\ 1 & 7 & 4 \end{bmatrix}$$

a) Aplicar una escalamiento (+2 píxeles) a la matriz **M**

$$M' = \begin{bmatrix} 3 & 3 & 6 \\ 2 & 5 & 5 \\ 1 & 7 & 4 \end{bmatrix} * \begin{bmatrix} \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \end{bmatrix} = \begin{bmatrix} \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \end{bmatrix}$$

b) Graficar la imagen resultante (**M'**) considerando el tamaño de los píxeles.





Universidad Católica de Santa María
Escuela Profesional de Ingeniería de Sistemas

14. Ordena el **pseudocódigo** para la siguiente **suma de matrices**:

$$B = \begin{pmatrix} b_{11} & \dots & b_{1n} \\ b_{21} & \dots & b_{2n} \\ \vdots & \ddots & \vdots \\ b_{m1} & \dots & b_{mn} \end{pmatrix} \quad A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \quad S_{i,j} = (a+b)_{ij} = a_{ij} + b_{ij}$$

Nombre del Algoritmo: SumaMatrices(A,B)

Declaración de variables :

A = Matriz A

m = número de filas

n = número de columnas

B = Matriz B

S = Matriz Resultante

Inicio (ordenar y transcribir las sentencias)

(1) Leer A y B

() _____

() _____

() _____

() _____

() _____

Fin

Sentencias

1. Leer A y B
2. fin Desde j
3. Desde j = 0 hasta n, incrementa j en 1, hacer:
4. Desde i = 0 hasta m, incrementa i en 1, hacer
5. fin Desde i
6. S(i,j) = A(i,j) + B(i,j)

Observación :El orden presentado no es correcto

a) **Caso de prueba 1:** Dado dos matrices del mismo tamaño(3x3) se debe obtener una matriz resultante al realizar una suma de estas dos matrices. Completar los dos siguientes caso de prueba como en el ejemplo:

Ejemplo :

$$\text{SumaMatriz} \left(\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{pmatrix} \right) \rightarrow \begin{pmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{pmatrix}$$

$$\text{SumaMatriz} \left(\begin{pmatrix} \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \end{pmatrix}, \begin{pmatrix} \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \end{pmatrix} \right) \rightarrow \begin{pmatrix} \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \end{pmatrix}$$



Universidad Católica de Santa María
Escuela Profesional de Ingeniería de Sistemas

15. Dada una matriz cuadrada A 4×4 (habitualmente nos referimos a ella a una matriz de orden 4) en donde los casos de prueba (A) Y (B) identifican un elemento en una posición determinada:

$$\text{Buscar}A_{03} \left(\begin{bmatrix} 12 & 3 & 2 & 33 \\ 5 & 4 & 1 & 14 \\ 8 & 13 & 27 & 37 \\ 7 & 12 & 21 & 19 \end{bmatrix} \right) \rightarrow 33 \quad \text{Buscar}A_{13} \left(\begin{bmatrix} 12 & 3 & 2 & 33 \\ 5 & 4 & 1 & 14 \\ 8 & 13 & 27 & 37 \\ 7 & 12 & 21 & 19 \end{bmatrix} \right) \rightarrow 14$$

(A) (B)

- a) Diseña los casos de prueba (C) y (D) como (A) o (B), para una búsqueda genérica **Buscar(M,e)** donde **M** es la matriz y **e** el elemento a buscar.

$$\text{Buscar}A_{___} \left(\begin{bmatrix} ______ \\ ______ \\ ______ \\ ______ \end{bmatrix} \right) \rightarrow ______$$

- b) Diseña el **algoritmo** para realizar la búsqueda genérica

Nombre del Algoritmo : Buscar(M,e)

Declaración de variables :

M = Matriz

m = Longitud (número de filas)

n = Ancho (número de columnas)

E = elemento a buscar

Inicio

Fin



Universidad Católica de Santa María
Escuela Profesional de Ingeniería de Sistemas

16. Ordena y completa el **pseudocódigo** para la siguiente **multiplicación de matrices** :

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mp} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{p1} & b_{p2} & \dots & b_{pn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1n} \\ c_{21} & c_{22} & \dots & c_{2n} \\ \dots & \dots & \dots & \dots \\ c_{m1} & c_{m2} & \dots & c_{mn} \end{bmatrix}$$

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + \dots + a_{1p}b_{p1}$$

a) Dado dos matrices **A(m x p)** y **B(p x n)**. Obtener la **matriz resultante** de la multiplicación de A y B. Diseña el siguiente **caso de prueba (2)** con valores aleatorios en las matrices y su respectiva matriz resultante:

(1) MultiplicaMatrices $\left(\begin{bmatrix} 3 & 7 & 5 \\ 9 & 4 & 11 \end{bmatrix}, \begin{bmatrix} 3 & 4 \\ 2 & 14 \\ 8 & 12 \end{bmatrix} \right) \rightarrow \begin{bmatrix} 63 & 170 \\ 123 & 224 \end{bmatrix}$

(2) MultiplicaMatrices $\left(\begin{bmatrix} \square & \square & \square \\ \square & \square & \square \end{bmatrix}, \begin{bmatrix} \square & \square \\ \square & \square \\ \square & \square \end{bmatrix} \right) \rightarrow \begin{bmatrix} \square & \square \\ \square & \square \end{bmatrix}$

b) Ordenar y transcribir el siguiente **Algoritmo** :

Declaración de variables :

A = Matriz A
B = Matriz B
m = número de filas de A
p = número de columnas de A y número de filas de B
n = número de columnas de B
C = Matriz Resultante

Inicio

(1) Leer A y B

() _____

() _____

() _____

() _____

() _____

() _____

() _____

Fin

Sentencias:

1. Leer A y B
2. fin Desde k
3. Desde k=1 hasta p, incrementa k en 1, hacer:
4. fin Desde j
5. fin Desde i
6. Desde i=1 hasta m, incrementa i en 1, hacer :
7. Desde j=1 hasta n, incrementa j en 1, hacer:
8. C(i, j) = p(i, j) + a (i, k) * b(k, j)

Observación : El orden presentado no es correcto

ANEXO B.

Encuesta de satisfacción

Cuestionario

Código: _____

1) Has participado de la actividad (Representación de imágenes) usando el juego:

- Si
- No

2) Si la respuesta es “**Si**”, ¿Qué apreciación tienes sobre el juego para aprender matrices?

3) ¿Crees que otros cursos deberían enseñarse utilizando videojuegos?

- Totalmente de acuerdo
- Regularmente
- No necesariamente
- No para nada

4) ¿Que cosas **no** te agradan de la carrera Ingeniería de sistemas y de los cursos de programación?

5) ¿Que cosas de la carrera Ingeniería de sistemas te agradan y cuales crees que deberían mejorar ?

ANEXO C.

Derechos de autor: Método de enseñanza



Universidad Católica de Santa María
Escuela Profesional de Ingeniería de Sistemas

Método de enseñanza de representación de imágenes de forma lúdica

1. Resumen

Se proporciona un software y un método de enseñanza que introduce de forma lúdica el aprendizaje de matrices así mismo su aplicación en representación de imágenes, el método se compone de ejercicios matemáticos los cuales están organizados en niveles de aprendizaje con una dificultad gradual, así como el uso intermedios lúdicos entre los ejercicios los cuales no exigen un conocimiento matemático pero sí exigen habilidades de juego y reacciones rápidas por parte del jugador, el cual mantiene un nivel de concentración del estudiante en el juego. Por último la inmersión competitiva en el uso de rankings del estudiante con el resto de su clase aumenta el nivel de motivación e interés en el estudiante durante su aprendizaje.

2. Campo de la invención

La presente invención hace referencia al campo de software educativo. Particularmente la invención presentada está orientada como juego digital educativo para estudiantes de ingeniería de sistemas.

3. Antecedentes de la invención

El uso de material interactivo para la enseñanza de matrices como diapositivas, videos, clases dinámicas así como juegos para su enseñanza son muchas veces los recursos utilizados como material de apoyo el la enseñanza por ejemplo:

Tradicionalmente a los estudiantes que llevan cursos de matemáticas y programación se suele enseñar la teoría pocas veces con el uso de material dinámico, en la parte práctica

se suele presentar ejercicios en clase, algunas veces ejercicios dinámicos (actividades como conversatorios) con la participación de los alumnos hasta incluso se utilizan juegos digitales como material de apoyo.

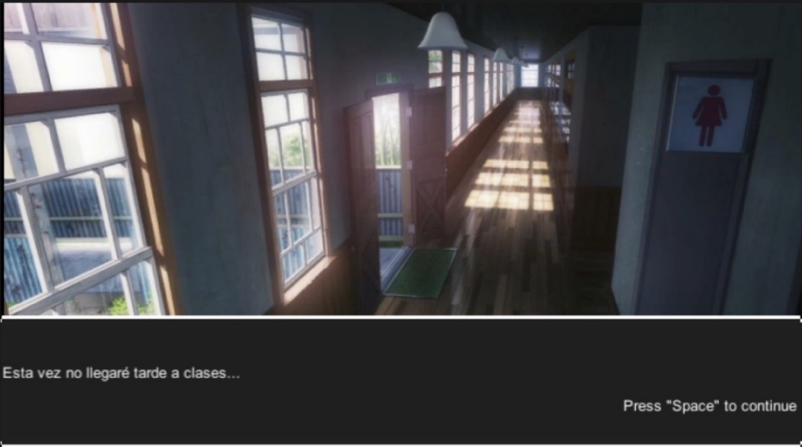
Estos juegos digitales normalmente no son muy efectivos en cursos de nivel universitario, existen técnicas de aprendizaje que mejoran la enseñanza en estudiantes de ingenierías las cuales tienen definido una serie de pasos para utilizarlas. Estas difieren de acuerdo al docente que las aplica en clase. Algunos juegos educativos son muy limitados por lo que si el docente desearía modificar o mejorar hasta incluso introducir alguna técnica en particular el software no se lo permite.

4. Breve descripción de la invención

El método implementado consiste en introducir de forma lúdica a los estudiantes, simular una actividad de su realidad en el juego (por ejemplo el asistir a clases) ayuda a sentirse identificado con el juego, una vez logrado esto es necesario identificar los retos que se le presentarán al jugador los cuales se irán presentando conforme el estudiante avance en los niveles. Por ejemplo utilizar puertas con cerraduras las cuales tienen que ser descifradas y resueltas para continuar con el juego son ideales para aplicar un reto (en el caso de representación de imágenes: El uso de operaciones matriciales para obtener una solución en transformaciones geométricas), sin embargo el abuso de estas ocasionan aburrimiento y desinterés en su uso por lo que es necesario diseñar un espacio entre retos (En el caso de representación de imágenes: puertas con cerraduras), los cuales deben estar relacionados con la historia del juego como por ejemplo: Correr y saltar para esquivar trampas, recolectar monedas y recompensas ayuda a que el jugador mantenga un estado emocional de motivación e interés alto en el juego. Además la inmersión del estudiante en un ambiente competitivo (poder visualizar el avance y la puntuación de otro estudiante) aumenta el interés y compromiso con el juego en culminar correctamente los retos planteados.

ANEXO D.

Guía del juego: Engineers Escaping



Esta vez no llegaré tarde a clases...

Press "Space" to continue

INTRODUCCIÓN

El Comienzo

Al principio verás la historia del juego, todo inicia al llegar a clases, encontrarás una grabadora con un audio, al reproducirla escucharás que los ingenieros han sido secuestrados y que si no los rescatas los obligarán a desaprobarte. (Presiona la **barra espaciadora** para avanzar más rápido en el flujo de mensajes)

1

INGRESA TUS DATOS

Codigo|UCSM

START ▶

INTRODUCCIÓN

LOGIN

Después de contar la historia tendrás que ingresar el **código universitario** de 8 dígitos para poder continuar con el juego

2

GAMEPLAY

CLOSE(X)




CAP.1 EL COMIENZO

GAME PLAY

Empezarás en la calle, tu personaje dirá que acaba de llegar a la ubicación indicada en el audio, a continuación te mostrará **el gameplay** del juego (las teclas que debes usar para moverte y activar los controles en el juego)


3



CAP.1 EL COMIENZO
MAIN STREET

Camina hasta la puerta principal, automáticamente te indicará que presiones la tecla "E" para entrar.


4



CAP.1 EL COMIENZO
PUERTA PRINCIPAL

En esta parte te hablará Zero quien dice haber estudiado las cerraduras de las puertas en esta casa.

5



PISTA RECIBIDA!

BASIC LOCK


Toma esto puede ser de ayuda...

Press "Space" to continue

CAP.1 EL COMIENZO
PUERTA PRINCIPAL

Zero te entregará la primera pista "Basic Lock", existen otras que Zero dejó dentro la casa, las cuales deberás encontrar en el juego.

7



PISTA RECIBIDA!

BASIC LOCK

Toma esto puede ser de ayuda...

Press "Space" to continue

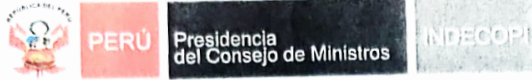


CAP.1 EL COMIENZO
PUERTA PRINCIPAL

Zero te entregará la primera pista "Basic Lock", existen otras que Zero dejó dentro la casa, las cuales deberás encontrar en el juego.

6

ANEXO E.

Resolución Derechos de autor EEW

 <p>DIRECCIÓN DE DERECHO DE AUTOR</p> <p>CERTIFICADO DE REGISTRO DE PROGRAMAS DE ORDENADOR (SOFTWARE)</p>		<p>Nro. Partida Registral: 01072- 2018 Asiento: 01 Fecha Presentación: 2018-05-28 Fecha de Inscripción: 2018-07-10 No. de Expediente: 001236-2018</p>
<p>DATOS DE LA OBRA</p> <p>Título: ENGINEERS ESCAPING WALKTHROUGH - EEW INEDITA Tipo de Obra: ORIGINARIA, Pais de Origen: PERU</p>		
<p>DATOS DEL AUTOR O AUTORES</p> <p>Apellidos y Nombres: HUAYNASI ARONE, ROBERTO Doc. de Identidad 74136639 Pais de Nacimiento: PERU CARLOS Domicilio: CALLE INDEPENDENCIA N° 108, PAUCARPATA, AREQUIPA. AREQUIPA Fecha de Nacimiento: 1995-07-19 Apellidos y Nombres: ROJAS GOMEZ, CHRISTIAN EMILIO Doc. de Identidad 73612755 Pais de Nacimiento: PERU Domicilio: CALLE SALAVERRY N° 205, CERRO COLORADO, AREQUIPA. AREQUIPA Fecha de Nacimiento: 1995-05-26 Apellidos y Nombres: CASTRO GUTIERREZ, EVELING Doc. de Identidad 29695284 Pais de Nacimiento: PERU GLORIA Domicilio: AV. PROGRESO N° 525, HUARANGUILLO, SACHACA, AREQUIPA. AREQUIPA Fecha de Nacimiento: 1971-11-28</p>		
<p>DATOS DEL PRODUCTOR</p> <p>Apellidos y Nombres: UNIVERSIDAD CATOLICA DE SANTA MARIA Domicilio: SAMUEL VELARDE N° 320, UMACOLLO, AREQUIPA, AREQUIPA. AREQUIPA</p>		
<p>DATOS DEL TITULAR O TITULARES</p> <p>Apellidos y Nombres: UNIVERSIDAD CATOLICA DE SANTA MARIA Doc. de Identidad 20141637941 Pais de Nacimiento: PERU Domicilio: SAMUEL VELARDE N° 320, UMACOLLO, AREQUIPA, AREQUIPA. AREQUIPA</p>		
<p>OBSERVACIONES : NINGUNA.</p>		
<p> DANIEL LAZO BARRETO Dirección de Derecho de Autor INDECOPI</p>		<p> FAUSTO VIENRICH ENRIQUEZ Director de Derecho de Autor INDECOPI</p>
<p>El derecho de autor protege exclusivamente la forma original y creativa, mediante la cual las ideas del autor son descritas, explicadas, ilustradas o incorporadas a las obras. No son objeto de protección las ideas contenidas en las obras literarias y artísticas, o el contenido ideológico o técnico de las obras científicas, ni su aprovechamiento industrial o comercial (artículos 8° y 9° del Decreto Legislativo Nro. 822.)</p>		

INSTITUTO NACIONAL DE DEFENSA DE LA COMPETENCIA Y DE LA PROTECCIÓN DE LA PROPIEDAD INTELECTUAL
 Calle De la Prosa 104, San Borja, Lima 41 - Perú / Telf.: 224 7800
 e-mail: consultas@indecopi.gob.pe / Web: www.indecopi.gob.pe



Universidad Católica de Santa María

☎ (51 54) 382038 Fax:(51 54) 251213 ✉ ucsm@ucsm.edu.pe 🌐 http://www.ucsm.edu.pe Apartado:1350

"IN SCIENTIA ET FIDE EST FORTITUDO NOSTRA"
(En la ciencia y en la fe está nuestra fortaleza)

SE EXPRESA RECONOCIMIENTO Y FELICITACION A LOS AUTORES DE LOS SOFTWARE "CLASH OF CLASH COFC" Y "ENGINEERS ESCAPING WALKTHROUGH - EEW"..-

RESOLUCION No. 25850 -R-2018

Arequipa, 2018 setiembre 17

Visto el Of. No. 1127-2018-VRINV mediante el cual el Sr. Vice Rector de Investigación solicita se emita Resolución de felicitación a los autores del diseño de los software "Engineers Escaping Walkthrough - EEW" y "Clash of Clash COFC", cuyo registro se ha logrado en INDECOPI a favor de la UCSM;

CONSIDERANDO:

Que, con Resolución No. 24162-R-2017 se aprobó el Contrato de Adjudicación de Investigación por el cual la UCSM es titular del Proyecto de Investigación "Construcción de Estrategias de Enseñanza-Aprendizaje basadas en PX (Player Experience), UX (User Experience) y TDL (Test-Driven Learning) para evitar la deserción de Ingeniería de Sistemas de la UCSM, celebrado entre el Equipo Investigador y la Institución;

Que, el Sr. Vice Rector de Investigación informa que producto de la investigación realizada por el equipo investigador se diseñó los software "Engineers Escaping Walkthrough - EEW" y "Clash of Clash COFC" que permiten a través de videojuegos, establecer estrategias de enseñanza aprendizaje, procediéndose a realizar los trámites pertinentes y logrando, en consecuencia, la inscripción de los mencionados software ante el Registro Nacional de Derechos de Autor y Derechos Conexos a favor de la UCSM;

Que, en tal sentido, el Sr. Vice Rector de Investigación manifiesta que la UCSM al estar como Productor y Titular de los Software antes mencionados y siendo de importancia y relevancia para la Institución es que solicita se expide una Resolución de reconocimiento y felicitación en favor de los autores de los indicados softwares, quienes son egresados de la E.P. de Ingeniería de Sistemas Sres. Roberto Huaynasi Arone y Christian Emilio Rojas Gómez, y fueron asesorados por la Jefe de Prácticas Eveling Gloria Castro Gutiérrez;

Que, en tal sentido, debe procesarse a emitir la Resolución correspondiente;

De conformidad con lo dispuesto por el Art. 89, inc. c) del Estatuto de la Universidad;

SE RESUELVE:

PRIMERO

Expresar el reconocimiento y felicitación a los autores de los software que a continuación se indica, por las razones expuestas en la parte considerativa de esta Resolución:

- "CLASH OF CLASH COFC" (Resolución de Registro No. 01145-2018/DDA-INDECOPI)
- Sr. Christian Emilio Rojas Gómez, egresado de la E.P. de Ingeniería de Sistemas
 - Sr. Roberto Huaynasi Arone, egresado de la E.P. de Ingeniería de Sistemas
 - Mgter. Eveling Gloria Castro Gutiérrez, Jefe de Prácticas Nombrada de la UCSM

./.





Universidad Católica de Santa María

(51 54) 382038 Fax:(51 54) 251213 ✉ucsm@ucsm.edu.pe 🌐http://www.ucsm.edu.pe Apartado:1350

AREQUIPA PERU

RESOLUCION No. 25850 -R-2018

- 2

./.
"ENGINEERS ESCAPING WALKTHROUGH - EEW" (Resolución de Registro No. 01146-2018/DDA-INDECOPI)

- Sr. Christian Emilio Rojas Gómez, egresado de la E.P. de Ingeniería de Sistemas
- Sr. Roberto Huaynasi Arone, egresado de la E.P. de Ingeniería de Sistemas
- Mgter. Eveling Gloria Castro Gutiérrez, Jefe de Prácticas Nombrada de la UCSM



SEGUNDO

El Vice Rectorado de Investigación, se encargará del cumplimiento e implementación de la presente.

Regístrese y comuníquese.



DR. MANUEL ALBERTO BRICEÑO ORTEGA
RECTOR
UNIVERSIDAD CATÓLICA DE SANTA MARÍA