Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

ANÁLISIS DE DIFERENTES LENGUAJES Y HERRAMIENTAS PARA EL DESARROLLO DE VIDEOJUEGOS

Carlos Andrés Barrios González
Asesorado por el Ing. Luis Fernando Quiñónez López

Guatemala, junio de 2011

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ANÁLISIS DE DIFERENTES LENGUAJES Y HERRAMIENTAS PARA EL DESARROLLO DE VIDEOJUEGOS

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

CARLOS ANDRÉS BARRIOS GONZÁLEZ
ASESORADO POR EL ING. LUIS FERNANDO QUIÑÓNEZ LÓPEZ

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, JUNIO DE 2011

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Juan Carlos Molina Jiménez
VOCAL V	Br. Mario Maldonado Muralles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Edgar Santos
EXAMINADOR	Ing. Pedro Pablo Hernández
EXAMINADOR	Ing. César Fernández
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

ANÁLISIS DE DIFERENTES LENGUAJES Y HERRAMIENTAS PARA EL DESARROLLO DE VIDEOJUEGOS

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha diciembre de 2009.

Carlos Andrés Barrios González

Guatemala, 20 de diciembre de 2010

Ingeniero
Carlos Alfredo Azurdia Morales
Coordinador
Comisión de Aprobación y Revisión de Tesis
Escuela de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Estimado Ingeniero:

Por medio de la presente hago de su conocimiento que he considerado el trabajo de tesis titulado **Análisis de Diferentes Lenguajes y Herramientas para el Desarrollo de Videojuegos**, que el estudiante Carlos Andrés Barrios González ha desarrollado como proyecto de graduación para optar al título de Ingeniero en Ciencias y Sistemas. Este trabajo ha sido de mi interés y a mi consideración el contenido del trabajo es satisfactorio y está finalizado en un 100%.

Como Asesor de este trabajo me hago responsable de la calidad y del contenido que la tesis ha alcanzado.

Sin otro particular y agradeciendo la oportunidad de colaboración a la educación e investigación universitaria que me da, me despido.

Atentamente,

LUIS QUIÑÓNEZ
INGENIERO EN C.C. Y SISTEMAS
COLEGIADO No. 7514

Ing. Luis Fernando Quiñónez López
Ingeniero en Ciencias y Sistemas
Asesor de Trabajo de Graduación
Colegiado No. 7514

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 26 de Enero de 2011

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **CARLOS ANDRES BARRIOS GONZALEZ** carné **2004-12861,** titulado: **"ANALISIS DE DIFERENTES LENGUAJES Y HERRAMIENTAS PARA EL DESARROLLO DE VIDEOJUEGOS",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, de trabajo de graduación titulado ***"ANÁLISIS DE DIFERENTES LENGUAJES Y HERRAMIENTAS PARA EL DESARROLLO DE VIDEOJUEGOS"****, presentado por el estudiante CARLOS ANDRÉS BARRIOS GONZÁLEZ, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

Ing. Marlon Antonio Pérez Turk
Director, Escuela de Ingeniería Ciencias y Sistemas

Guatemala, 15 de junio 2011

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG.197-2011

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **ANÁLISIS DE DIFERENTES LENGUAJES Y HERRAMIENTAS PARA EL DESARROLLO DE VIDEOJUEGOS,** presentado por el estudiante universitario, **Carlos Andrés Barrios González,** autoriza la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
DECANO

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, junio de 2011

/cc
c.c. archivo.

ACTO QUE DEDICO A:

MIS PADRES Hary Anabella González y Andrés Barrios, por su tiempo y apoyo durante todo este tiempo.

MI HERMANO José Rodrigo Barrios, por ser una gran parte de mi vida.

AGRADECIMIENTOS A:

MIS PADRES	Hary Anabella González y Andrés Barrios, por apoyo y cariño siempre.
MI HERMANO	José Rodrigo Barrios, por su apoyo incondicional en todos los momentos de mi vida.
MI ABUELA, MIS TÍOS Y PRIMOS	Por su inmenso cariño y apoyo incondicional en todos los momentos de mi vida.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES VII
GLOSARIO XI
RESUMEN XVII
OBJETIVOS XIX
INTRODUCCIÓN XXI

1. MARCO TEÓRICO 1
    1.1. Los videojuegos 1
    1.2. Breve historia de los videojuegos 3
    1.3. Generaciones de consolas 12
    1.4. Clasificación y tipos de videojuegos 15
    1.5. Géneros principales 17
        1.5.1. Videojuego de plataformas 17
        1.5.2. Videojuego de disparos 18
            1.5.2.1. Disparos en primera persona o FPS 18
            1.5.2.2. Disparos en tercera persona o TPS 18
            1.5.2.3. *Shoot 'em up* 19
            1.5.2.4. Sigilo 19
        1.5.3. *Beat 'em up* 20
        1.5.4. Videojuegos de lucha 21
        1.5.5. Videojuegos de *arcade* 22
        1.5.6. Videojuegos de simulación 23
        1.5.7. Videojuego de estrategia 24
        1.5.8. Videojuegos educativos 25
        1.5.9. Videojuego de rol 25

1.5.10. Videojuego de aventura 26
1.6. Los videojugadores o *gamers* 27
1.7. Programadores de videojuegos 28
1.8. Proceso de desarrollo 29
1.8.1. Pre-producción 29
1.8.2. Producción 31
1.8.3. Pruebas 31
1.8.4. *Milestones* 32
1.8.5. Mantenimiento 32
1.9. Desarrollo de videojuegos independiente 33
1.9.1. IGF - *Independent Games Festival* 34
1.9.2. Ejemplos 35
1.9.2.1. *World of* Goo 35
1.9.2.2. *Braid* 37
1.9.2.3. Aquaria 39
1.10. Lenguajes y herramientas 40
1.10.1. APIs y bibliotecas 41
1.10.1.1. DirectX 42
1.10.1.2. OpenGL 46
1.10.2. Otras herramientas 48
1.10.3. Microsoft XNA 49
1.10.3.1. XNA *Framework* 49
1.10.3.2. XNA *Game Studio* 50

2. ANÁLISIS Y SELECIÓN DE LAS HERRAMIENTAS PARA LA IMPLEMENTACIÓN DEL VIDEOJUEGO 51
2.1. Breve descripción 51
2.2. Selección del lenguaje de programación 52
2.2.1. Lenguaje de programación 52

2.3. Historia del lenguaje 52
2.4. Características del lenguaje 54
2.5. Selección de compiladores 54
2.5.1. BennuGD 54
2.5.1.1. Características 57
2.5.2. Gemix *Studio* 58
2.5.2.1. Características 59
2.6. Otras herramientas utilizadas 60
2.6.1. *Noteptad++ for* FENIX/BENNU V5.03 60
2.6.2. FPG *Edit* V0.12 61
2.6.2.1. Características 61
2.6.3. FNT *Edit* V0.3.285 63
2.6.4. PakAtor 64
2.6.5. *Adobe® Illustrator® CS4* 65
2.6.6. *Adobe® Flash® CS4* 66
2.6.7. GIMP 67
2.6.8. *Map Editor* 68
2.6.9. *Tiled Map Editor* 69
2.6.10. *Guitar Pro* 70
2.6.11. *TuxGuitar* 71
2.6.12. FL *Studio* 9 72
2.6.13. *MadTracker* 2 73
2.7. Resumen 74
2.7.1. Compiladores 74
2.7.2. Herramientas 75

3. PROCESO DE DESARROLLO 77
3.1. Fase de Pre-Producción 77
3.1.1. Género 77

3.1.2. *Game play* 77
3.1.3. Mecánica 78
3.1.4. Controles y acciones 79
3.1.5. Reglas del ambiente 82
3.1.5.1. Ejemplo 83
3.1.6. Historia 84
3.1.7. *Story board* 85
3.1.8. Arte conceptual 89
3.2. Fase de Producción 95
3.2.1. Programación del videojuego 95
3.2.2. *Loop* principal del videojuego 95
3.2.3. Personaje principal 96
3.2.4. Proceso para detección de paredes 98
3.2.5. Proceso para detección del techo y el suelo 100
3.2.6. *Map tiler* y edición de *tile maps* 102
3.2.6.1. Algoritmo 102
3.2.6.2. Ejemplo 103
3.2.7. Edición de mapas utilizando *Tiled Map Editor* 104
3.2.8. Diseño gráfico 106
3.2.8.1. Animaciones 106
3.2.8.2. Niveles y otros elementos 108
3.2.9. Edición de efectos de sonido y música 110
3.2.9.1. Escritura de la partitura 110
3.2.10. Fase de pruebas 113
3.2.10.1. Pruebas *alpha* 114
3.2.10.2. Pruebas beta 114
3.2.10.3. *Betatesters* oficiales 114
3.2.11. Publicación 115
3.2.11.1. Página del videojuego 115

3.2.11.2. Enlaces de descarga 116
3.2.11.3. Información de contacto 116
3.3. Resultados finales 117
3.3.1. Evaluación de los usuarios 117
3.3.2. Tablas y gráficos 118
3.3.3. Jugabilidad 119
3.3.4. Promedio de resultados 120
3.4. *Screenshots* 121

CONCLUSIONES 125
RECOMENDACIONES 127
REFERENCIAS 129
BIBLIOGRAFÍA 133

ÍNDICE DE ILUSTRACIONES

FIGURAS

1. OXO 3
2. *Tennis for Two* 4
3. *Spacewar* en el ordenador PDP-1 5
4. Ralph Baer y el Magnavox *Odyssey* 5
5. Commodore 64 *BASIC* V2.0 7
6. Mario Bros para NES 8
7. iPhone de *Apple* 11
8. Sonic *the Hedgehog* 17
9. MDK 19
10. *Splinter Cell Mobile Game* 19
11. *God of War* 20
12. *Street Fighter IV* 21
13. Pac-Man 22
14. *Roller Coaster Tycoon 2* 23
15. *Age of Empires 2* 24
16. *Brain Challenge* 25
17. *Warcraft 3* 26
18. *Monkey Island* 26
19. Compilador Gemix en Linux Ubuntu 28
20. IGF 34
21. *World of* Goo de *2D Boy* 35
22. Pantallas del videojuego *World of* Goo 36
23. *Braid* 37

24. *Braid* de Jonathan Blow 38
25. Aquaria de *Bit Blot* 39
26. Logo de Microsoft DirectX 42
27. Microsoft DirectX *Developer* 44
28. Microsoft XNA *Game Developer* 45
29. Logo oficial de OpenGL 47
30. Microsoft XNA 49
31. Microsoft XNA *Game Studio* 50
32. Dr. Topo 51
33. Div *Games Studio* 2 de *Hammer Technologies* 53
34. Bennu *Game Development* 55
35. Extensión Bennu3D 56
36. Gemix *Studio* 58
37. *Notepad++ for* Fenix/Bennu 60
38. FPG *Edit* 61
39. FNT *Edit* 63
40. PakAtor 64
41. Ejemplos de las gráficas creadas para el videojuego Dr. Topo 65
42. Animación en *Adobe Ilustrator CS4* utilizada para los *sprites* 66
43. Edición de *sprites* utilizando GIMP 67
44. *Map Editor* de SkyGem *Software* 68
45. Edición de mapas para el videojuego en *Tiled Map Editor* 69
46. Edición de partituras en *Guitar Pro* para *Windows* 70
47. Edición de partituras en *TuxGuitar* para Linux 71
48. Entorno de trabajo de FL *Studio* 9 72
49. Edición de *tracks* en *MadTracker* 2 73
50. Prueba del primer nivel del videojuego 78
51. Controles y acciones para 79
52. Ejemplo de un nivel y su mapa de durezas 83

53. Dr. Topo en el primer nivel del videojuego 84
54. *Story board* básico del videojuego 85
55. Ideas para los personajes 87
56. Ideas para los niveles del videojuego 88
57. Arte conceptual generada a partir de codificación 89
58. Arte conceptual inspirada en la *demoscene* 90
59. Arte conceptual de los enemigos del videojuego 91
60. Arte conceptual de los niveles del videojuego 93
61. Arte conceptual del videojuego 94
62. Durezas en el nivel de prueba 100
63. Durezas de suelo y techo 102
64. Matriz y nivel generado por el algoritmo del *map tiler* 103
65. Carga de *tiles* en *Tiled Map Editor* 104
66. Edición de niveles utilizando *Tiled Map Editor* 105
67. Animación del personaje principal 106
68. Edición de las animaciones 107
69. Edición de niveles 108
70. Edición elementos para los niveles 109
71. Edición de partitura 110
72. Partitura de tema tema del videojuego 111
73. Edición de pistas en FL *Studio* 112
74. Versión *alpha* de Dr. Topo 113
75. Página principal del videojuego 115
76. Página de descargas 116
77. Gráfica con los resultados de la sección “Gráficos” 118
78. Gráfica con los resultados de la sección “Jugabilidad” 119
79. Gráfica con los resultados de los promedios totales 120
80. Imágenes del videojuego publicado 121
81. *Screenshots* del videojuego Dr. Topo en Linux Ubuntu 9.10 124

TABLAS

I. Generaciones de consolas 12
II. Versiones de los compiladores utilizados para el videojuego 74
III. Herramientas utilizadas para la creación del videojuego 75
IV. Acciones básicas del personaje principal 80
V. Colores de durezas 82
VI. Códigos de dureza para las paredes 98
VII. Códigos de dureza para el suelo y el techo 100
VIII. Datos recolectados de los usuarios 117
IX. Promedio de resultados de la sección “Gráficos” 118
X. Promedio de resultados de la sección "Jugabilidad” 119
XI. Promedio de resultados totales 120

GLOSARIO

API Un API o *Application Programming Interface* (Interfaz de Programación de Aplicaciones) es el conjunto de especificaciones e implementación de comunicación entre componentes de *software* que ofrece una biblioteca como una capa de abstracción.

BennuGD Lenguaje de programación orientado a procesos distribuido bajo licencia GNU *General Public License.* Es un *fork* del proyecto fénix desarrollado por SplinterGU.

Consola Sistema electrónico de entretenimiento interactivo que permite la ejecución de juegos electrónicos almacenados en dispositivos de almacenamiento secundario.

DAW Una Estación de Audio Digital (*Digital Audio Workstation*) es un sistema computarizado que permite la generación de sonidos sintetizados, grabación y edición de pistas de audio.

Demoscene	Subcultura informática especializada en la producción de demos audiovisuales ejecutados en tiempo real en una computadora. Surge con el auge de las computadoras de 8 y 16 *bits*.
DirectX	Colección de API para Microsoft *Windows* creada para facilitar complejas tareas relacionadas con aspectos multimedia.
Div	Lenguaje de programación orientado a procesos que tiene una estructura basada en el lenguaje de programación C. Creado por Daniel Navarro y desarrollado para la creación de juegos en el entorno de MS-DOS.
Dxi	Componente de *software* que puede ser cargado como *plugin* en una aplicación para procesamiento en tiempo real de audio y funcionar como un sintetizador virtual.
Fenix	Proyecto de *software* libre para la creación de un compilador alternativo al lenguaje de programación DIV.
Fnt	Archivo de fuente de caracteres utilizado originalmente por Div *Games Studio*.
Fork	Bifurcación de un proyecto de *software* que toma una dirección diferente al proyecto ya existente.

FPG	Archivo de colección de gráficos utilizado por Div *Games Studio*.
Game Engine	Sistema de *software* diseñado para la creación y desarrollo de videojuegos.
Gemix	Herramienta para desarrollo de videojuegos totalmente compatible con el lenguaje de programación DIV.
GNU	Licencia creada por *Free Software Foundation* que busca principalmente proteger la distribución, uso y modificación de *software* libre.
Handheld	Término que describe a un ordenador portátil que puede ser trasladado a cualquier parte mientras es utilizado.
Indie	Término que hace referencia a los videojuegos desarrollados por un grupo pequeño de personas sin el apoyo de una empresa distribuidora.
Map Tiler	Aplicación que permite la construcción de mapas de grandes dimensiones a partir de *tiles* de menor tamaño.

MIDI	Siglas de Interfaz Digital de Instrumentos Musicales. Es un estándar de comunicación serial que permite a los sintetizadores, secuenciadores, controladores y otros dispositivos electrónicos comunicarse y compartir información.
Milestone	Hitos que representan metas internas del desarrollo que delimitan tiempos de entrega del proyecto.
OpenGL	Especificación estándar desarrollada originalmente por *Silicon Graphics Inc*. que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D.
Raster	Imagen matricial que representa una rejilla rectangular de píxeles que pueden ser visualizados en algún dispositivo de representación.
Renderizado	Término que hace referencia al proceso de generación de una imagen desde un modelo.
Scroll	Acción de desplazamiento horizontal o vertical en un videojuego 2D en el que pueden existir distintos planos para generar efecto de profundidad.

SDL	Conjunto de bibliotecas multiplataforma desarrolladas en lenguaje de programación C que proporcionan funciones básicas para la manipulación de objetos en 2D, gestión de imágenes y manejo de sonido para el desarrollo de aplicaciones multimedia y videojuegos.
Sketch	Dibujo o bosquejo que representa algún elemento del videojuego.
Softsynth	Sintetizadores basados en *software* para la generación de audio digital que son usualmente de menor costo que el *hardware* dedicado.
Sprite	Tipo de mapa de *bits* dibujado en la pantalla del ordenador por *hardware* especializado que no necesita cálculos adicionales del procesador.
Tablatura	Forma de escritura musical específica para ciertos instrumentos que difiere de la notación de escritura musical común.
Tile	Un *tile* o baldosa es un elemento gráfico del videojuego que puede ser utilizado para la construcción de un fondo o mapa de gran tamaño.

Tracker	Herramienta de *software* que funciona como secuenciador utilizando *samples* o muestras digitales en distintos canales. Las notas musicales son generalmente representadas por caracteres alfanuméricos y códigos hexadecimales.
VST	Interfaz desarrollada por Steinberg que permite la interconexión de sintetizadores, sistemas de grabación, *plugins* y editores de audio que reemplaza el *hardware* tradicional de grabación.

RESUMEN

Un videojuego es un programa informático cuya finalidad es la entretención de los usuarios que interactúan con un dispositivo electrónico a través de una interfaz de usuario capaz de generar una respuesta visual, sonora y de otros tipos. Estos programas recrean entornos virtuales en los que el jugador puede controlar a un personaje y otros elementos del entorno.

En los inicios de la historia de los juegos electrónicos, un programador de videojuegos era la persona encargada de realizar todo el proceso de creación, incluyendo el diseño, la programación y el arte. Más que programadores eran artistas a tiempo completo.

A medida de que las capacidades de *hardware* fueron mejorando, este proceso se especializó y se dividió en personas que se encargaban de realizar la historia y trama del videojuego, artistas gráficos, encargados de sonido, etc. Esta división de labores provocó la separación de la disciplina de programación de videojuegos del diseño de éstos.

En los años 90 la distribución de videojuegos comerciales era manejada por grandes compañías distribuidoras y los desarrolladores independientes se vieron forzados a crear su propia empresa distribuidora o a encontrar alguna compañía dispuesta a la distribución de sus programas. Años después, con el surgimiento de las tiendas en línea fue posible vender sus videojuegos al mercado mundial con una mínima inversión inicial.

El desarrollo de videojuegos independientes o *Indie Games* es el proceso de creación de juegos electrónicos sin el apoyo de una empresa distribuidora. Estos videojuegos son generalmente desarrollados por un solo individuo o por un pequeño grupo de personas y el proceso de creación completo puede llegar a durar desde unos pocos días hasta varios años dependiendo de la complejidad del proyecto.

La creación de un videojuego es independiente del lenguaje de programación, aunque dependiendo del lenguaje y su paradigma existen distintas ventajas y desventajas al momento de desarrollarlo. La mayoría de juegos comerciales son escritos principalmente en C, C++ y lenguaje ensamblador debido a las complejas limitantes del *hardware* de las consolas. En el caso de los videojuegos para computadora personal se utilizan bibliotecas y APIs como DirectX, OpenGL y SDL.

OBJETIVOS

General

Realizar un análisis y descripción de distintas herramientas y lenguajes de programación para el desarrollo de videojuegos examinando las características, ventajas y desventajas que pueden facilitar el proceso de selección de estos elementos antes de iniciar la realización de un videojuego.

Específicos

1. Documentar el proceso de desarrollo de videojuegos, su historia, etapas, herramientas, bibliotecas, lenguajes y otros elementos relacionados con el desarrollo de videojuegos independientes.
2. Redactar un informe final con la información y conclusiones recopiladas a lo largo de la investigación.
3. Analizar y evaluar las herramientas utilizadas durante el proceso de implementación de un videojuego de plataformas.
4. Implementar un videojuego de plataformas con desplazamiento lateral que ejemplifique el proceso de desarrollo de videojuegos independientes en lenguaje de programación Div utilizando el compilador Gemix.

INTRODUCCIÓN

El negocio de los videojuegos ha evolucionado hasta convertirse en algo más que un producto informático. Se ha convertido un fenómeno social, un objeto de investigación que gracias a los avances tecnológicos sobre los que se desarrollan y funcionan alcanzan un nivel de realismo e interactividad inimaginable hace algunas décadas.

Los videojuegos cuentan con una muy exitosa industria que en los últimos años ha generado más dinero que la del cine. La industria de videojuegos está compuesta por diferentes productos que se ejecutan sobre distintas plataformas como consolas, computadoras personales, dispositivos portátiles, teléfonos móviles y juegos en línea.

La creación de un videojuego es independiente del lenguaje de programación, aunque dependiendo del lenguaje y su paradigma existen distintas ventajas y desventajas al momento de desarrollarlo. Las herramientas y bibliotecas soportadas por el lenguaje pueden facilitar en gran medida la creación del programa al simular leyes físicas, conectividad a través de red, soporte de motores o *engines* 3D, etc (Vida Mombiela, y otros, 2007). La selección del compilador es de gran importancia ya que algunos permiten generar programas ejecutables para distintas plataformas y consolas por lo que debe de tenerse en cuenta antes de iniciar el proyecto.

La tecnología por sí misma, como una biblioteca para el desarrollo de videojuegos, no es suficiente para crear un juego. Cualquier persona, incluso un programador de RPG II y Cobol puede escribir uno (Harbour, 2002).

La creación de un videojuego es en principio un arte que puede ser expresado de muchas maneras, depende de la creatividad e innovación del desarrollador para utilizar las herramientas existentes y crear las que necesite para lograr sumergir a los jugadores en una realidad alterna, un universo que solamente se encuentra limitado por la creatividad del desarrollador (Fernández Ureña, 2002).

1. MARCO TEÓRICO

1.1. Los videojuegos

Un videojuego es un programa informático cuya finalidad es la entretención de uno o varios usuarios que interactúan con un dispositivo electrónico a través de una interfaz capaz de generar una respuesta visual, sonora y de otros tipos. Los dispositivos electrónicos que ejecutan el programa pueden ser computadoras personales, consolas, dispositivos portátiles y juegos en línea.

Estos programas recrean entornos virtuales en los que el jugador puede controlar a un personaje y otros elementos del entorno para conseguir uno o varios objetivos por medio de reglas determinadas.

Las interacciones entre los jugadores y el aparato electrónico en el que se ejecuta el videojuego se realizan a través de dispositivos de salida como un monitor, un televisor o un proyector y de dispositivos de entrada como el teclado, *mouse*, *joystick*, *gamepad*, detectores de movimiento, volantes, etc. que dependen de cada tipo de plataforma. El dispositivo electrónico que ejecuta el videojuego puede ser una computadora personal, un artefacto especialmente creado con este fin como las videoconsolas y las máquinas de *arcade* o algún otro dispositivo capaz de ejecutar programas como los teléfonos móviles, dispositivos *handheld*, etc.

El juego normalmente está grabado en un dispositivo de almacenamiento que puede ser de diversos tipos tales como: cartuchos, discos magnéticos, CD, DVD, discos Blu-ray, HD-DVD, etc.

A lo largo de sus más de 30 años de evolución, los videojuegos han ido incorporando las características y capacidades de las nuevas tecnologías como la combinación de varios lenguajes audiovisuales en un mismo soporte, la interactividad, la capacidad para procesar información y la conectividad. Todo ello, explorando las posibilidades de este nuevo medio para ofrecer experiencias lúdicas de gran valor a sus jugadores.

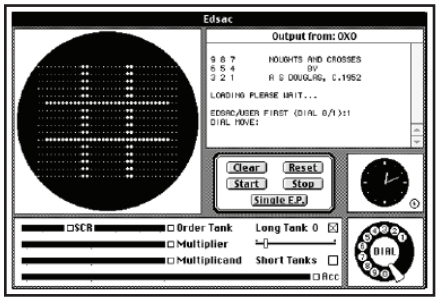
Es importante establecer una diferencia entre el soporte y el contenido del videojuego. Los soportes son las máquinas electrónicas que permiten ejecutar el contenido o programa del videojuego pudiendo ser aparatos diseñados específicamente para esta funcionalidad como las consolas domésticas y las consolas portátiles que responden a criterios de funcionalidad que permiten al usuario interactuar con el videojuego. También es posible utilizar videojuegos en computadoras portátiles y dispositivos móviles que aunque no han sido diseñados con tal finalidad, permiten utilizar el videojuego a través de elementos propios como el *mouse*, el teclado numérico, etc.

1.2. Breve historia de los videojuegos

El inicio de la historia de los videojuegos puede situarse alrededor del año 1947, cuando Thomas T. Goldsmith Jr. y Estle Ray llenaron una aplicación de patente en Estados Unidos el 25 de enero de 1947. Habían creado un sistema electrónico de juego que simulaba el lanzamiento de misiles contra un objetivo y se basaba en las pantallas de radar que usaba el ejército en la entonces reciente segunda guerra mundial. Este sistema funcionaba con válvulas y usaba una pantalla de rayos catódicos. Permitía ajustar la velocidad y la curva del disparo pero no contaba con movimiento de video en la pantalla por lo que no se le considera un videojuego como tal.

En cambio, el inicio del entretenimiento digital puede situarse en el primer simulador de vuelo diseñado en los Estados Unidos para el entrenamiento de pilotos en los años cuarenta. No es posible señalar cuál fue el primer videojuego debido a las distintas definiciones que se han establecido a lo largo del tiempo aunque es posible considerar como primer videojuego el *Nought and Crosses* también conocido como OXO, desarrollado en 1952 por Alexander Douglas. Consistía en una versión computarizada del juego tres en raya que se ejecutaba sobre el computador EDSAC y permitía que un jugador humano se enfrentara a la computadora.

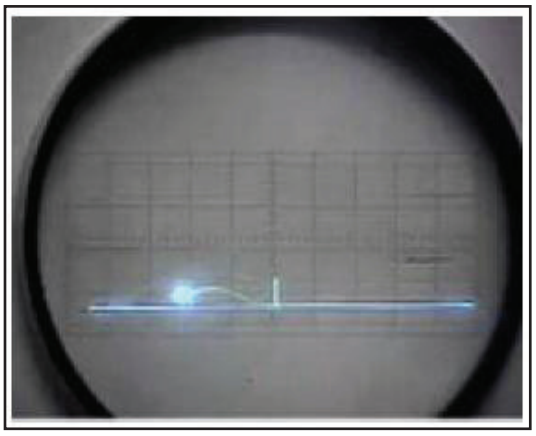
Figura 1. **OXO**



Fuente: Pixfans. http://www.pixfans.com/wp-content/uploads/2007/10/oxo.png.

En 1957, William Higinbotham crea *Tennis For Two*, un simulador de tenis de mesa que utilizaba como monitor un osciloscopio del Brookhaven *National Laboratory*. Este simulador utilizaba una vista lateral en la que se observaba una pista con una red que separaba los campos de cada jugador. Contaba con un controlador compuesto por un pulsador que permitía golpear la pelota virtual y un mando analógico que controlaba la dirección de ésta.

Figura 2. ***Tennis for Two***



Fuente: Wildgames. http://wildgames.es/wp-content/uploads/2008/10/tennis_for_two.jpg.

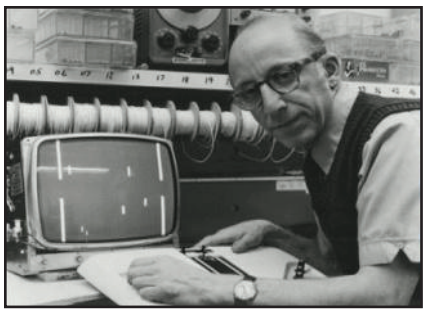
Otro de los primeros videojuegos fue *Spacewar*. Este es el primer videojuego interactivo para ordenador creado alrededor del año 1961 por el estudiante Steve Rusell junto a otros estudiantes del Instituto de Tecnología de Massachusetts (MIT) en el ordenador PDP-1 que contaba con una pequeña pantalla de rayos catódicos volviéndose muy popular entre las universidades aunque sin ningún éxito al comercializarse. Este juego utilizaba gráficos vectoriales en el que dos jugadores podían controlar la velocidad y dirección de dos naves espaciales que luchaban entre sí.

Figura 3. ***Spacewar* en el ordenador PDP-1**

Fuente: Joi Ito. http://es.wikipedia.org/wiki/Spacewar!.

En 1966 Ralph Baer, Albert Maricon y Ted Dabney iniciaron el desarrollo del proyecto *Fox and Hounds* dando inicio a los videojuegos de uso doméstico. Este proyecto se convertiría en el primer sistema doméstico de videojuegos, el Magnavox *Odyssey* lanzado en 1972 que permitía utilizar juegos pregrabados y conectarse a un televisor. Durante este año se convirtió rápidamente en un éxito de ventas logrando ser la primera consola comercial de la historia al vender alrededor de 100,000 unidades con un costo de $100 por unidad.

Figura 4. **Ralph Baer y el Magnavox *Odyssey***



Fuente: Olds School Generation. http://oldschoolgeneration.blogspot.com/2010/04/magnavox-odyssey-la-primera-consola-de.html.

Este contaba con un *hardware* muy reducido careciendo de procesador y memoria. Estaba compuesta solamente por transistores, condensadores y resistencias.

En 1972 Nolan Bushnell crea Pong, un videojuego en dos dimensiones que simulaba un tenis de mesa. Atari se encargó de comercializarlo y apareció en las primeras máquinas de *arcade* convirtiéndose en el juego más famoso y popular de esa época. Tres años después Atari crea su primera consola doméstica con el nombre insignia del juego Atari Pong que solamente permitía utilizar este único videojuego. Esta idea no fue muy exitosa ni rentable ya que no permitía otra posibilidad. Entonces, en 1977 Atari consigue el éxito con su nueva consola la Atari 2600, toda una revolución hasta entonces en el mundo de los primeros videojuegos.

La Atari 2600 contaba con la innovación de poder cambiar de juegos mediante un sistema de cartuchos y sólo poseía 8 *bits* de potencia. En esta época, Atari contaba con un grupo de desarrollo para la investigación de sistemas de videojuegos de última generación denominado *Cyan Engineering* que trabajó en el prototipo Stella. Este sistema contaba con una CPU completa y tres circuitos integrados dedicados al manejo de gráficos, sonidos, manejo de memoria y control de entrada y salida. Esta combinación de circuitos permitía un enorme potencial para su tiempo.

Durante muchos años gracias a su amplio catálogo de juegos, sus ganancias permitieron a la compañía incluso comprar licencias de películas impulsando su éxito que duró más de una década hasta que finalmente aparece Nintendo en el mercado de las consolas.

A principios de los años 80 aparecen los ordenadores personales. Es la época de los ordenadores Amstrad. En 1980 también aparece el *Odyssey* 2 de Phillips y el sistema Intellivision de Mattel. Durante esta época destacan los videojuegos Pac-Man de NAMCO *y Battle Zone* de Atari que fue la base para programas de entrenamiento militar utilizado por el Ejército de los EE.UU.

El ordenador Commodore 64 es lanzado en 1982 y contaba con un intérprete de lenguaje *BASIC*. También permitía utilizar lenguajes de programación como COBOL por medio de cartuchos de expansión. Poseía una excelente arquitectura de *hardware* y potentes capacidades gráficas y sonoras lo que permitió que se creara gran cantidad de videojuegos y aplicaciones para este ordenador. Durante este año también fue fabricado el Sinclair ZX *Spectrum*.

Figura 5. **Commodore 64 *BASIC* V2.0**

Fuente: Comodore 64 BASIC V2.0. http://es.wikipedia.org/wiki/Archivo:C64_startup_animiert.gif.

En 1981 aparece el famoso *Donkey* Kong de Nintendo, diseñado por Shigeru Miyamoto. Surge por primera vez uno de los personajes más duraderos y conocidos de la historia de los videojuegos: Mario, aunque entonces era un carpintero llamado *Jumpman*.

Figura 6. **Mario Bros para NES**

Fuente: SMB1, Nintendo. http://en.wikipedia.org/wiki/File:NES_Super_Mario_Bros.png.

En 1985 sale al mercado la consola NES o Famicom en Japón, respaldada por la genialidad de Shigeru Miyamoto en la creación de sus juegos convirtiéndose en la primera videoconsola exitosa para el fabricante. Es considerada la consola más exitosa de su época y contribuyó a revitalizar significativamente la industria estadounidense de videojuegos que había sufrido la llamada crisis del videojuego.

A partir de esta consola la empresa Nintendo estableció un modelo de negocios estandarizado en la época contemporánea referente a la licencia de *software* para desarrolladores tipo *third-party* (Sanchez, 2003).

A medida que la tecnología fue progresando, los desarrolladores de videojuegos descubrieron que los nuevos microprocesadores de bajo costo ofrecían una nueva vía para crear juegos más sofisticados; las plataformas de *hardware* programable podían ejecutar una gran variedad de juegos y no solamente uno. Esto provocó que eventualmente las consolas con cartuchos se volvieran populares (Clinton, 2010).

A finales de la década de los 80, SEGA presenta su primer equipo de 16 *bits* conocido como Sega *Genesis* en América y Sega *Mega Drive* en Europa y Asia. Presenta una evolución en los entornos gráficos 2D al permitir más detalles en los escenarios y personajes.

En los años 90 se da el salto técnico a la generación de 16 *bits* con la *Super* Famicon o Snes de Nintendo, la PC *Engine* de Nec, la CPS *Charger* de Capcom y la *Mega Drive* de Sega. Se inicia la creación de videojuegos en entornos tridimensionales para computadoras personales con juegos como Woolfstein, *Doom* y *Alone in the Dark*.

También aparece la Neo Geo de SNK que igualaba las capacidades y prestaciones técnicas de las máquinas de *arcade,* pero con un precio muy alto como para llegar de forma masiva a los hogares de los jugadores. Se lanza el videojuego *Virtual Racing*, el primer juego con gráficos poligonales para consola. Posteriormente se introduce el uso del CD-ROM ya que permitía mayor capacidad de almacenamiento y resultaba más económico que los cartuchos.

Con el surgimiento de videojuegos en 3D se inicia la generación de 32 *bits* con consolas como la *PlayStation* de Sony y la generación de 64 *bits* con consolas como Atari Jaguar de Atari y Nintendo 64 de Nintendo.

Aparece el *Dreamcast* en 1998, siendo la última videoconsola producida por Sega. Contaba con 128 *bits* y fue desarrollada en cooperación por Hitachi y Microsoft. Era una consola potente capaz de ejecutar gráficos muy superiores a los de cualquier sistema de videojuegos de esa época. Tenía un precio accesible, un buen catálogo de juegos, posibilidad de jugar *on line* al contar con un módem y cuatro puertos para conectar varios mandos que venían de serie.

En 2001, Microsoft ingresa a la industria de las consolas creando la Xbox y su juego estelar Halo. Nintendo lanza la *Gamecube* y Sega se da cuenta de que no puede competir especialmente contra la nueva máquina de Sony. Sega anuncia que descontinuara la *Dreamcast* y que ya no produciría *hardware*, convirtiéndose solo en desarrolladora de *software* desde el año 2002.

Aunque el ordenador personal es la plataforma más cara de videojuegos es también la que permite mayor flexibilidad. Su flexibilidad proviene del hecho de que es posible añadir componentes que se pueden mejorar constantemente, como tarjetas gráficas, tarjetas de sonido, accesorios, mandos, etc. Además es posible actualizar los juegos con parches oficiales o con nuevos añadidos realizados por la compañía que creó el juego o por otros usuarios.

Nokia entra al mercado de las consolas portátiles con el N-Gage, un híbrido de teléfono y consola portátil en 2003. Gran cantidad de teléfonos móviles incluyen soporte para Java ME permitiendo que se cree gran cantidad de juegos para estos dispositivos.

En 2004 se presenta la Nintendo DS y la *PlayStation Portable*. Durante el año 2006, Sony presenta su *PlayStation* 3 y Nintendo la Wii, una consola desarrollada en colaboración con ATI e IBM. Su característica más distintiva es su mando inalámbrico o Wiimote el cual cuenta con la detección de la aceleración de los movimientos en tres dimensiones.

En 2007, *Apple* presenta sus nuevos iPod *touch* y iPhone que pueden funcionar como consolas de videojuegos al desarrollar una plataforma beneficiada en gran parte por la tecnología multitáctil.

Figura 7. **iPhone de *Apple***

Fuente: Movilerevolutions. http://movilevolutions.com/wp-content/uploads/2010/08/iphone-game.jpg.

Los videojuegos fueron los precursores de la invasión de la computadora casera y constituyen una de las industrias más rentables de todos los tiempos (Foreste, 1992). Al generalizarse el uso de ordenadores personales, surgen las primeras empresas dedicadas al emergente mercado de los videojuegos y gracias a los avances tecnológicos que los soportan la calidad gráfica, sonora, el realismo y la interactividad se hacen cada vez mas fantásticos convirtiendo a los videojuegos en algo más que un producto informático sino también en un negocio, un objeto de investigación y un fenómeno social generalizado.

1.3. Generaciones de consolas

Tabla I. **Generaciones de consolas**

GENERACIÓN	*BITS*	PERÍODO	CONSOLAS
Primera	2	1972-1977	Magnavox *Odyssey* Magnavox *Odyssey* 100 Magnavox *Odyssey* 200 Atari Pong Coleco Telestar
Segunda	3	1976-1984	Fairchild *Channel* F Atari 2600 Videopac G7000 Intellivision Atari 5200 Vectrex Arcadia 2001 ColecoVision TV-*Game* 6 Sega SG-1000

Continúa tabla I

Tercera	4	1983-1992	Atari 7800 Nintendo *Entertainment System* (NES) *GameBoy* Sega *Master System* *GameGear* PV-1000 Epoch *Cassette Vision* *Supergame* VG 3000
Cuarta	16	1988-1996	Sega *Mega Drive* Neo-Geo *Super* Nintendo *Entertainment System* (SNES) TurboGrafx-16/PC *Engine* CD-TV CD-i
Quinta	32 y 64	1993-2002	3DO AmigaCD32 Atari *Jaguar* Sega *Saturn* *Virtual Boy* *PlayStation* Nintendo 64 *Apple* Pippin

Continúa tabla I

			Casio *Loopy* Neo Geo CD PC-FX Playdia FM *Towns* Marty
Sexta	128	1998-2006	*Sega Dreamcast* *PlayStation* 2 Xbox Nintendo *GameCube* *GameBoy Advance*
Séptima	256	2006- Actualidad	Wii Xbox 360 *Playstation* 3 Nintendo DS Nintendo DSi *PlayStation Portable* Zeebo

Fuente: Wikipedia.org. http://es.wikipedia.org/wiki/Videoconsola.

1.4. Clasificación y tipos de videojuegos

Es posible clasificar a los videojuegos de acuerdo a diversos criterios. Algunos son fáciles de catalogar dado que se les puede situar claramente en un determinado grupo pero otros pueden ser considerados mixtos, es decir, que pertenecen a dos o más grupos, por lo que su clasificación es más ambigua y difícil.

En 1984, Chris Crawford (Goldstein, 1983) hace una clasificación de los videojuegos en tres grandes categorías a partir del tipo de estrategias implicadas:

- ✓ Juegos de acción y destreza
- ✓ Juegos de estrategia
- ✓ Juegos cognitivos

Martín (Martín, 1995), clasifican los videojuegos en siete tipos distintos a partir de las características generales del desarrollo del juego:

- ✓ *Arcade*
- ✓ Aventura
- ✓ Estrategia
- ✓ Juegos de rol
- ✓ Simuladores
- ✓ Educativos
- ✓ Juegos de mesa

La FAD (FAD, 2002), en su investigación utiliza la siguiente clasificación de los tipos de videojuegos:

- ✓ Juegos de plataforma
- ✓ Aventura gráfica
- ✓ De rol
- ✓ Simuladores
- ✓ De práctica de algún deporte
- ✓ De estrategia deportiva
- ✓ De estrategia no deportiva
- ✓ De disparo
- ✓ De lucha

Al existir gran cantidad de géneros y distintas clasificaciones, se presenta una gran dificultad para delimitar de una forma clara y exacta la clasificación de cada videojuego por lo que la mayor parte de los títulos se clasifican como mixtos y pertenecen a más de un género.

1.5. Géneros principales

1.5.1. Videojuego de plataformas

Los videojuegos de plataformas se caracterizan por tener un protagonista que debe de recorrer, saltar o escalar una serie de plataformas, obstáculos y enemigos mientras se recolectan objetos para poder completar el objetivo del videojuego. Suelen usar desplazamiento lateral o *scroll*. Entre estos se encuentran:

- ✓ *Super* Mario Bros
- ✓ Sonic *The Hedgehog*
- ✓ *Prince of Persia*
- ✓ *Super* Mario 64

Figura 8. **Sonic *the Hedgehog***

Fuente: Sonic The Hedgehog, SEGA. http://sonamylove.files.wordpress.com/2011/02/sonic-the-hedgehog-211.jpg.

1.5.2. Videojuego de disparos

En estos videojuegos el protagonista debe de lograr un objetivo a base de disparos. Las acciones básicas del juego consisten en mover al personaje y utilizar un arma. Generalmente estos videojuegos no poseen un guión trabajado y estructurado pero destacan en calidad gráfica y la jugabilidad. La mecánica del juego impone habitualmente al jugador tener buenos reflejos y precisión. Algunos videojuegos de este género son:

- ✓ *Doom*
- ✓ *Quake*
- ✓ *Half Life*
- ✓ Halo
- ✓ *Counter Strike*
- ✓ Max Payne
- ✓ *Metal Gear Solid*

Dentro de este género es posible encontrar:

1.5.2.1. Disparos en primera persona o FPS

Perspectiva en la que el personaje es visto desde atrás y da la sensación de ser el personaje.

1.5.2.2. Disparos en tercera persona o TPS

El personaje es visto desde atrás o con una perspectiva isométrica.

Figura 9. **MDK**

Fuente: MDK, Shiny Entertainment. http://www.gamespot.com/pc/action/mdk/index.html.

1.5.2.3. *Shoot 'em up*

Basados en el uso continuo de armas y el enfrentamiento con jefes al final de cada misión del juego.

1.5.2.4. Sigilo

Se basan en la estrategia y sigilo en vez de la confrontación directa con los enemigos.

Figura 10. ***Splinter Cell Mobile Game***

Fuente: Splinter Cell, Gameloft. http://www.gameloft.com/mobile/splinter-cell-conviction/.

1.5.3. *Beat 'em up*

Estos videojuegos son también llamados videojuegos de lucha a progresión. Son similares a los juegos de lucha, con la diferencia de que los jugadores deben combatir con un gran número de enemigos mientras avanzan a lo largo de varios niveles. Algunos ejemplos son:

- ✓ *Double Dragon*
- ✓ *Battletoads*
- ✓ *Final Fight*
- ✓ *Streets of Rage*
- ✓ *Devil May Cry*
- ✓ *God of War*

Figura 11. ***God of War***

Fuente: God of War, SCEA. www.godofwar.com/es_ES/index.htm.

1.5.4. Videojuegos de lucha

Este género recrea combates y peleas entre personajes controlados por un jugador o por la computadora. Los personajes son vistos desde una perspectiva lateral, como si se tratase de un espectador. Tienen énfasis en las artes marciales reales u otros tipos de enfrentamientos sin armas como el boxeo o la lucha libre. Algunos ejemplos son:

- ✓ *Street Fighter*
- ✓ *Mortal Kombat*
- ✓ *Fatal Fury*
- ✓ Tekken
- ✓ *Soul Calibur*

Figura 12. ***Street Fighter IV***

Fuente: Street Fighter IV, Capcom. http://www.streetfighter.com/us/ssfiv.

1.5.5. Videojuegos de *arcade*

Género que fue muy popular durante los años 80. Se caracteriza por la simplicidad y acción rápida de jugabilidad. No requiere de trama en la historia, solo son juegos largos o repetitivos. Entre estos se encuentran:

- ✓ *Street Fighter*
- ✓ Pac-Man
- ✓ *Space Invaders*
- ✓ *Asteroids*
- ✓ *Missile Command*
- ✓ *Galaxian*

Figura 13. **Pac-Man**

Fuente: Pac-Man, Namco. http://vicbengames.blogspot.com/2011/02/retro-analisis-pac-man.html.

1.5.6. Videojuegos de simulación

Este género se caracteriza por llevar un aspecto de la vida real a un videojuego en el que se tiene control de lo sucede. En algunas de sus subcategorias, se toma un concepto o una situación y se deja que el usuario explore las distintas opciones como la simulación de construcción, administración de parques de diversiones o la creación de familias y ciudades. Algunos de los videojuegos de este género son: The Sims, Sim *City*, *Roller Coaster Tycoon*, *Flight Simulator*, etc.

Entre las subcategorias de este género se encuentran:

- ✓ Simulación de combate
- ✓ Simulación de construcción y administración
- ✓ Simulación de vida
- ✓ Simulación de citas
- ✓ Simulaciones médicas
- ✓ Simulación de vehículos
- ✓ Simulación de fotografías
- ✓ Simulación de deportes

Figura 14. ***Roller Coaster Tycoon 2***

Fuente: Roller Coaster Tycoon 2, Chris Sawyer. http://en.wikipedia.org/wiki/File: Rct2_magicmountain.jpg.

1.5.7. Videojuego de estrategia

Se caracterizan por la necesidad de manipular a un numeroso grupo de personajes, objetos o datos para lograr los objetivos. Por su temática se pueden clasificar en:

- ✓ Estrategia bélica
- ✓ Estrategia de gestión

Mientras por su mecánica pueden ser clasificados en:

- ✓ Estrategia en tiempo real o RTS
- ✓ Estrategia por turnos o TBS

Algunos videojuegos de este género son:

- ✓ *Age of Empires*
- ✓ *StarCraft*
- ✓ *Warcraft*
- ✓ *Civilization*

Figura 15. ***Age of Empires 2***

Fuente: AoE II, Ensemble Studios. http://en.wikipedia.org/wiki/File:Age_ii_feudal_age_celts.jpg.

1.5.8. Videojuegos educativos

Tienen como objetivo proporcionar al usuario algún tipo de conocimiento. Son videojuegos que enseñan mientras divierten y entretienen a los jugadores. En algunos casos se duda de que sea un género de videojuego, ya que el concepto no está muy desarrollado.

Figura 16. ***Brain Challenge***

Fuente: Brain Challenge, Gameloft. http://en.wikipedia.org/wiki/File:Brainchallenge1.jpg.

1.5.9. Videojuego de rol

Videojuegos en los que el protagonista interpreta un papel y ha de mejorar sus habilidades mientras interactúa con el entorno y otros personajes. Estos se caracterizan por la interacción con el personaje, trama, historia profunda y evolución del personaje a medida que la historia avanza. Entre estos se encuentran:

- ✓ *The Elder Scroll IV*
- ✓ *Final Fantasy*
- ✓ *Dungeons & Dragons*
- ✓ *Warcraft*

Figura 17. ***Warcraft* 3**

Fuente: Warcraft III: Reign of Chaos, Blizzard Entertainment. http://playzet.com/juegos/warcraft-3-frozen-throne/.

1.5.10. Videojuego de aventura

Son videojuegos en los que el protagonista debe avanzar en la trama interactuando con diversos personajes y objetos. Algunos ejemplos son: *The Legend of Zelda*, *Maniac Mansion*, *Monkey Island* y Zork.

Figura 18. ***Monkey Island***

Fuente: Monkey Island, LucasArts. http://miburbujamisnormas.blogspot.com/2010/12/monkey-island.html.

1.6. Los videojugadores o *gamers*

Un *gamer* es la persona que utiliza los videojuegos completándolos parcial o totalmente y se caracteriza por tener gran cantidad de conocimiento sobre videojuegos. Una de las primeras investigaciones sobre quiénes utilizaban los videojuegos fue realizada en el año 1989 en Norteamérica por Provenzo (Provenzo, 1991). Provenzo situó la edad media de las personas usuarias mayoritarias de videojuegos entre los 8 y los 15 años de edad. A finales de 1992 un 90% de sujetos cuya edad comprendía un rango de 12 a 35 años reconocía haber jugado alguna vez.

Sanz, Maeso y Borreguero (Sanz, y otros, 2004) afirman que la edad de las personas que usan videojuegos se ha ampliado de 25 a 35 años, siendo 13 años la edad en la que se empieza a jugar. Actualmente se puede considerar que el universo de población que usa los videojuegos va desde los 6-10 años hasta los 24-30 años. El videojugador se divide en tres grupos:

- ✓ *Gamers*: grupo de jugadores experimentados que se caracterizan por llevar varios años jugando e invertir gran cantidad de horas utilizándolos.
- ✓ Videojugadores casuales: pertenece al grupo de jugadores no tradicionales que se caracterizan por ser usuarios relativamente nuevos.
- ✓ *Programmer*: videojugador profesional, que lucra participando en campeonatos y torneos oficiales, o trabajando para las compañías desarrolladoras como *testers* de errores en los videojuegos o contribuyendo con retroalimentación y críticas hacia el equipo desarrollador.

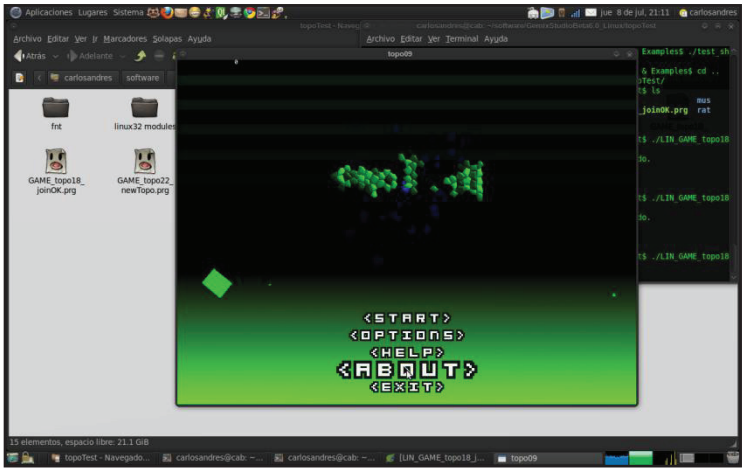
1.7. Programadores de videojuegos

En los primeros días de la historia de los juegos electrónicos, un programador de videojuegos era la persona encargada de realizar todo el proceso de creación, incluyendo el diseño, la programación y el arte. Más que programadores eran artista a tiempo completo.

Estos programadores solitarios, quienes diseñaban, programaban y realizaban el arte por si mismos han sido reemplazados por ejércitos de especialistas (Clinton, 2010).

A medida de que las capacidades de *hardware* fueron mejorando y las máquinas se hicieron más potentes, este proceso se especializó y se dividió en personas que se encargaban de realizar la historia y trama del videojuego, artistas gráficos, encargados de sonido, etc. Esta división de labores provocó la separación de la disciplina de programación de videojuegos del diseño de estos.

Figura 19. **Compilador Gemix en Linux Ubuntu**



Fuente: propia.

1.8. Proceso de desarrollo

El proceso de desarrollo de un videojuego es realizado por una sola persona o un grupo de personas que trabajan juntos y pueden ser parte de una gran organización. El desarrollo de un videojuego profesional usualmente inicia con el diseño del este e incluye las siguientes etapas:

1.8.1. Pre-producción

Normalmente antes de que el proyecto inicie, la idea debe de ser aprobada por la compañía desarrolladora o el publicista. Un videojuego difícilmente progresará sin el interés de una empresa interesada en publicarlo.

Durante esta etapa es necesario definir los aspectos fundamentales del videojuego, entre los que se encuentran:

- ✓ Género: dentro de que género o géneros se desarrollara el videojuego.
- ✓ *Game play*: se definen los elementos que generaran entretención al momento de utilizar el videojuego.
- ✓ *Story board*: conjunto de ideas sobre los personajes, el ambiente, sonidos, música, modelos 3D y otros elementos que serán parte del videojuego.

Los diseñadores del videojuego y el equipo de desarrollo producen un documento de diseño que describe los conceptos y elementos del juego en detalle. Estos documentos pueden incluir bocetos preliminares de distintos aspectos del videojuego.

Luego de esto, se detallan los elementos que compondrán el videojuego. Esto proporcionará una idea clara de todo el videojuego al grupo de desarrollo. Entre estos elementos se encuentran:

- ✓ Historia: define la historia y trama del juego en la que se desenvolverán los personajes.
- ✓ Arte conceptual: establece el aspecto general del videojuego. Los artistas se basarán en las ideas originales de los creadores para diseñar un conjunto de propuestas de como lucirá el videojuego.
- ✓ Mecánica: especifica el funcionamiento general del videojuego. Dependiendo del género se definen las interacciones y reglas que existen y rigen el funcionamiento de éste.
- ✓ Sonido: se describen los elementos sonoros necesarios para el videojuego como: efectos de sonido, música, voces y sonido ambiental.
- ✓ Diseño de programación: describe la manera en la que el videojuego será implementado mediante algún lenguaje de programación. Durante esta fase se pueden generar diagramas que describen el funcionamiento estático y dinámico del programa, las interacciones con los usuarios y diferentes estados que atravesará el videojuego como *software*.

Antes de que el proceso de aprobación del diseño esté completado, los artistas y programadores inician su trabajo desarrollando código rápido y sucio (*quick-and-dirty*) para presentar los prototipos a los interesados en el proyecto.

1.8.2. Producción

Para la producción de un videojuego usualmente se define un período de tiempo en el que el proyecto deberá de estar totalmente terminado. Los programadores se encargan de la escritura del código, los artistas crean las imágenes, *sprites* y modelos 3D de los elementos del videojuego, los ingenieros de sonido componen los efectos y música y los diseñadores de niveles crean los niveles y pantallas.

Durante este proceso el diseñador debe de modificar el documento de diseño para que refleje la visión actual del juego debido a que algunas características y niveles son añadidos o eliminados. Además debido a los ambientes dinámicos que posee el videojuego algunos aspectos pueden ir variando durante el tiempo. Los niveles posteriores son mucho más fáciles de crear y el tiempo se reduce hasta completar una visión clara del videojuego.

1.8.3. Pruebas

Las pruebas se inician luego de lograr que el programa sea jugable experimentando con algún nivel o parte del *software* por lo que habitualmente los *testers* trabajan en varios proyectos a la vez. Generalmente esta etapa se lleva a cabo en dos fases:

- ✓ Pruebas *alpha*: pruebas realizadas por un pequeño grupo de personas que están involucradas en el desarrollo del videojuego. Pueden incluirse: artistas, diseñadores, programadores, etc. Su propósito es corregir los defectos más graves y mejorar características de jugabilidad no contempladas durante el diseño.

- ✓ Pruebas beta: pruebas realizadas por un equipo externo de jugadores. Luego de estas pruebas el videojuego debe contener la menor cantidad posible de defectos menores y ningún defecto medio o crítico.

Los *tester* deben de ser parte del equipo, en lugar de encontrarse separados. Esto aumenta la velocidad de *feedback* a los desarrolladores sobre los posibles defectos (Clinton, 2010).

1.8.4. *Milestones*

Un videojuego comercial usualmente debe completar varios hitos que representan metas internas del proyecto y que delimitan tiempos de entrega. Entre los hitos se encuentra la realización de versiones con algún grupo de características definido.

1.8.5. Mantenimiento

Un videojuego para consola debe considerarse 100% completado antes de salir a la venta debido a que no puede ser cambiado posteriormente. Con la introducción de las consolas con acceso a internet como la Wii de Nintendo y el *PlayStation 3* de Sony es posible instalar parches a videojuegos que presentan errores y *bugs* como sucede con los juegos para computadora personal.

Debido a que pueden existir numerosos conflictos y problemas de compatibilidad con el *hardware* de las computadoras personales, muchos desarrolladores liberan parches luego de su publicación que pueden ser descargados generalmente desde el sitio web de la empresa desarrolladora.

1.9. Desarrollo de videojuegos independiente

El desarrollo de videojuegos independientes o *Indie Games*, es el proceso de creación de juegos electrónicos sin el apoyo de una empresa distribuidora. Estos videojuegos son generalmente desarrollados por un solo individuo o por un pequeño grupo de personas y el proceso de creación completo puede llegar a durar desde unos pocos días hasta varios años dependiendo de la complejidad del proyecto.

Hace más de cuatro décadas prácticamente no existía la industria de los videojuegos, los juegos electrónicos eran creados totalmente por programadores independientes. Estos programas generalmente eran distribuidos de persona a persona como *freeware* o *shareware*.

En los años 90, la distribución de videojuegos comerciales era manejada por grandes compañías distribuidoras y los desarrolladores independientes se vieron forzados a crear sus propias empresas distribuidoras o a encontrar alguna empresa dispuesta a la distribución de sus programas. Años después, con el surgimiento de las tiendas en línea fue posible vender sus videojuegos al mercado mundial con una mínima inversión inicial.

Recientemente, muchos desarrolladores independientes decidieron liberar el código fuente de sus videojuegos, aumentando el interés de potenciales participantes para su proyecto y posibilitando de esta forma lograr creaciones más complejas. Incluso han sido lanzadas adaptaciones para consolas como Xbox 360, Wii y *PlayStation 3*.

1.9.1. IGF - *Independent Games Festival*

Figura 20. **IGF**

Fuente: IGF, http://www.igf.com/.

En 1998, *Think Services* el productor de la revista *Game Developer*, Gamasutra.com y la *Game Developers Conference* estableció el Festival de Videojuegos Independiente para incentivar la innovación en el desarrollo de juegos y reconocer a los mejores desarrolladores independientes.

Esta competencia ha otorgado alrededor de $50,000 en premios a los mejores creadores independientes y a estudiantes en diferentes categorías buscando crear un evento similar al festival *Sundance* de cine independiente. Algunos de los premios que se otorgaron en el evento que se realizó en marzo de 2010 son:

- ✓ Seumas McNally *Grand Prize* ($20,000)
- ✓ *Excellence In Visual Art* ($2,500)
- ✓ *Excellence In Audio* ($2,500)
- ✓ *Excellence In Design* ($2,500)
- ✓ Nuovo *Award* ($2,500)
- ✓ *Technical Excellence* ($2,500)
- ✓ *Audience Award* ($2,500)
- ✓ IGF *Student Showcase Winner* (10 ganadores de $500).
- ✓ *Best Student Game* ($2,500)

1.9.2. Ejemplos

1.9.2.1. *World of* Goo

Videojuego independiente producido por *2D Boy*, un grupo de desarrollo formado por Kyle Gabler y Ron Carmel. Este videojuego fue nominado al premio Seumas McNally, el Premio de Diseño e Innovación y Excelencia Técnica en el Festival de Juegos Independientes.

Figura 21. ***World of* Goo de *2D Boy***

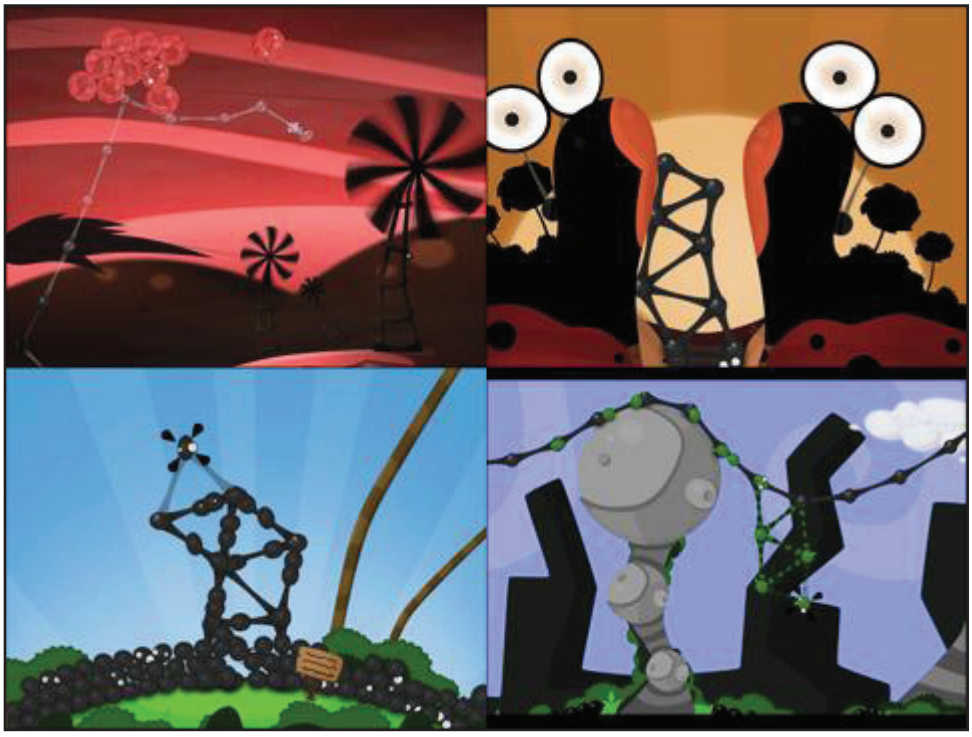
Fuente: World of Goo, http://www.worldofgoo.com/.

Fue lanzado para los sistemas operativos Microsoft *Windows*, Mac OS X y Linux. En el año 2008 también fue publicado para la consola Wii de Nintendo. Los desarrolladores estiman el coste de producción en unos 10.000 dólares, incluyendo alquiler, comida y equipo mínimo. El éxito del juego se atribuye a la presencia en la web del mismo, así como a los premios conseguidos en el Festival de Juegos Independientes en la Conferencia de Desarrolladores de Juegos en 2007, haciendo que las distribuidoras que anteriormente les habían ignorado, quisieran entonces publicar el título (Mysore).

En su desarrollo se emplearon un buen número de tecnologías de código abierto como *Simple* DirectMedia *Layer* (SDL)*, Open Dynamics Engine* y TinyXML. Subversion y Mantis *Bug Tracker* se utilizaron para el proceso de coordinación (Murphy).

También se involucró a la comunidad sobre la cual se delegó la traducción a neerlandés, francés, alemán, italiano y español para la versión europea que vio la luz en diciembre de 2008 (2D Boys).

Figura 22. **Pantallas del videojuego *World of* Goo**



Fuente: World of Goo, http://www.worldofgoo.com/.

1.9.2.2. *Braid*

Braid es un videojuego de plataformas desarrollado Jonathan Blow para el servicio Xbox *Live Arcade* de Xbox 360. El título fue presentado oficialmente durante la conferencia de prensa de Microsoft del *Tokyo Game Show* 2007 y salió a la venta el 6 de agosto de 2008 al precio de 1200 Microsoft *Points*. El título ha sido portado a *Windows*, Mac y *PlayStation* (Xbox.com).

Figura 23. ***Braid***

Fuente: Braid, http://www.braid-game.com/.

Su desarrollo duró aproximadamente tres años y cinco meses y el creador invirtió alrededor de $180,000. El desarrollo del juego parte de unas fuertes influencias literarias, "Las ciudades invisibles" de Italo Calvino, un conjunto de relatos cortos sobre distintas ciudades de ficción en diferentes realidades que funcionan de forma distinta y "Los sueños de Einstein" del físico Alan Lightman, en el que un Einstein que aún no tenía del todo clara la relatividad piensa en cómo puede comportarse el tiempo en el universo y la relación que eso tiene con lo que hay y como es la gente, si la hay, de dicho universo (Plunkett).

Durante el *Independent Games Festival* de 2006, *Braid* ganó el premio *Game Design* por su increíble trabajo artístico.

Figura 24. ***Braid* de Jonathan Blow**

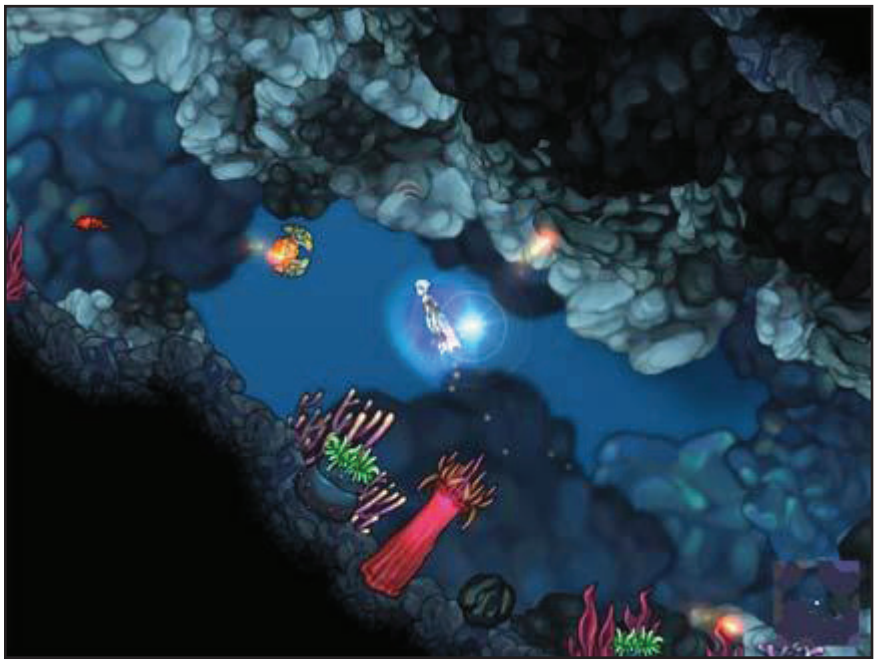
Fuente: Braid, http://www.braid-game.com/.

1.9.2.3. Aquaria

Aquaria es un videojuego de *scroll* lateral desarrollado por la compañía independiente *Bit Blot,* formada por Alec Holowka y Derec Yu. El proceso de desarrollo tomó más de dos años.

En el 2007 el videojuego ganó el premio Seumas McNally *Grand Prize* del festival IGF y fue finalista en las categorías de innovación, excelencia en arte visual y excelencia en audio.

Figura 25. **Aquaria de *Bit Blot***



Fuente: Aquaria, http://www.bit-blot.com/aquaria/.

1.10. Lenguajes y herramientas

La mayoría de juegos comerciales son escritos principalmente en C, C++ y lenguaje ensamblador debido a las complejas limitantes del *hardware* de las consolas. En el caso de los videojuegos para computadora personal se utilizan bibliotecas y APIs como DirectX, OpenGL, y SDL.

Luego de que el diseño inicial del videojuego ha sido aprobado es necesario que los desarrolladores decidan que lenguaje de programación utilizar. Esta selección depende de varios factores como:

- ✓ Lenguajes familiares y conocidos por el equipo de programadores
- ✓ Plataforma de destino
- ✓ Velocidad de ejecución necesaria para el juego
- ✓ Uso de *engines*
- ✓ Uso de APIs u otras bibliotecas

Actualmente, debido a las capacidades para compilación de binarios y su paradigma orientado a objetos, C++ es el lenguaje más popular y comúnmente utilizado para el desarrollo de videojuegos.

También existen varios lenguajes de *script* utilizados para la generación de inteligencia artificial y escenas gráficas que son interpretados en tiempo de ejecución. Esto permite generar contenido con un alto nivel de lógica a las condiciones del entorno en el videojuego resultando en un mayor realismo aunque en menores velocidades de ejecución.

El lenguaje de programación java es utilizado principalmente para la creación de videojuegos para plataformas y tecnologías móviles. Algunos teléfonos móviles permiten ejecutar juegos y aplicaciones basados en la tecnología *Adobe Flash* y su lenguaje *ActionScript*.

Lenguajes como Ada, Phyton, C# y Div tienen un muy pequeño impacto en la industria de desarrollo y son generalmente utilizados por programadores que crean juegos electrónicos como un pasatiempo. C# es popular para la creación de herramientas utilizadas para el desarrollo de videojuegos y es utilizado por XNA de Microsoft.

1.10.1. APIs y bibliotecas

Una decisión clave es cuales bibliotecas y APIs utilizar. Existe gran cantidad de bibliotecas que pueden facilitar las tareas al momento de programar el videojuego. Algunas bibliotecas pueden manejar el renderizado de las gráficas, procesar sonidos, procesar entradas, proveer rutinas de inteligencia artificial y facilitar la codificación de la lógica del juego.

Esta selección también depende de la plataforma en la que el programa deberá funcionar. Existen algunos *frameworks* y bibliotecas que permiten el desarrollo multiplataforma para que el programa pueda funcionar en varias plataformas distintas sin necesidad de hacer mayores cambios en el código proporcionando portabilidad.

Los APIs más populares para manejo de graficas 3D para Microsoft *Windows* son DirectX y OpenGL.

1.10.1.1. DirectX

Es una colección de API para Microsoft *Windows* creada para facilitar complejas tareas relacionadas con aspectos multimedia. Aunque esta desarrollado exclusivamente para el sistema operativo *Windows*, WineHQ es una implementación *open source* del API para sistemas Unix y Linux.

Figura 26. **Logo de Microsoft DirectX**

Fuente: Microsoft Corporation. http://www.sizzlingbit.com/2009/09/23/directx-11-windows-vista-sp2-download/.

DirectX esta formada por los siguientes APIs:

- ✓ Direct3D: utilizado para procesamiento y programación de gráficos en 3D.
- ✓ DirectGraphics: utilizado para manejo y dibujo de imágenes en 2D y para representación de imágenes en 3D.
- ✓ DirectInput: utilizado para procesar datos de entrada como: teclado, *mouse, joystick* y otros controles.
- ✓ DirectPlay: utilizado para comunicaciones en red.

- ✓ DirectSound: utilizado para la reproducción y grabación de sonidos de ondas.
- ✓ DirectMusic: utilizado para la reproducción de pistas musicales compuestas con DirectMusic *Producer*.
- ✓ DirectShow: utilizado para reproducir audio y vídeo con transparencia de red.
- ✓ DirectSetup: utilizado para la instalación de componentes DirectX.

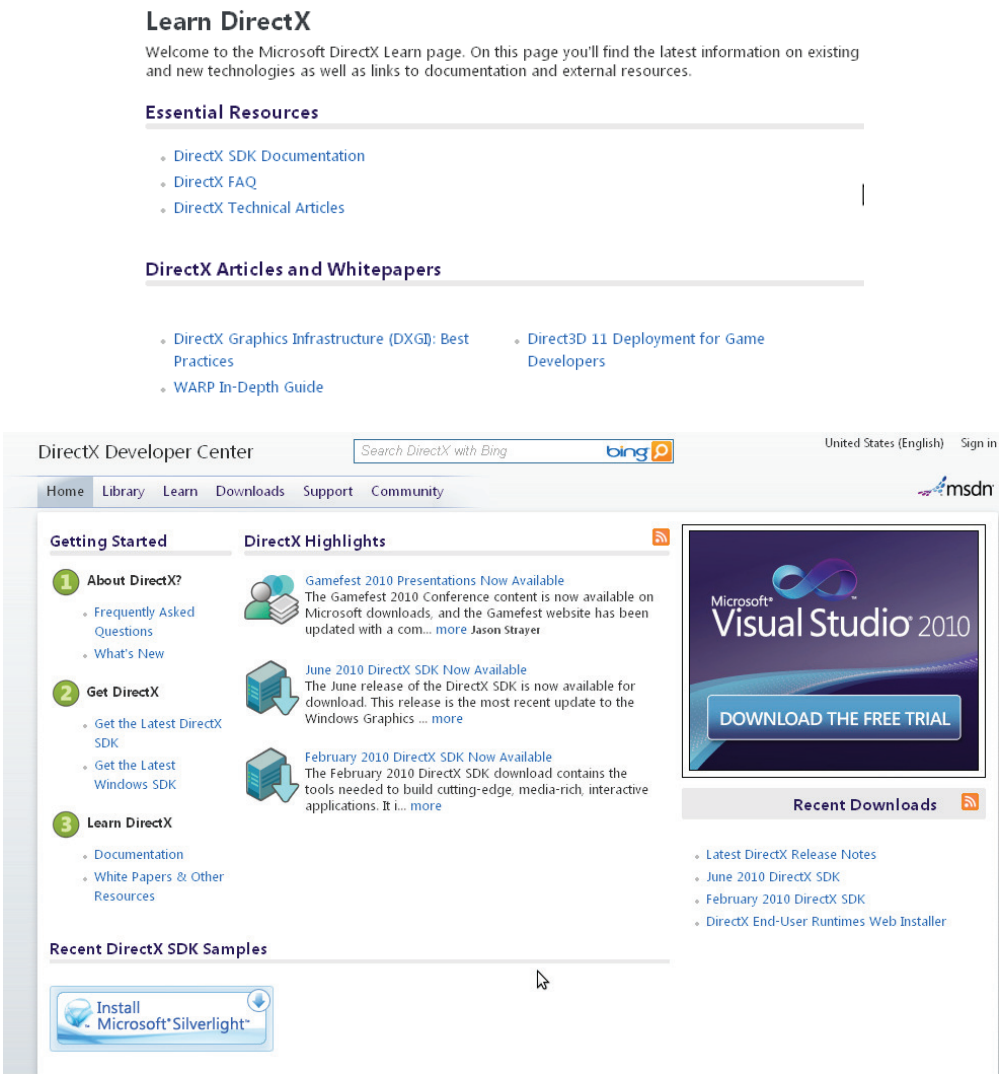
El *kit* de desarrollo de *software* (SDK) de Microsoft DirectX está compuesto por:

- ✓ Los componentes gráficos compuestos por Direct3D, la biblioteca de utilidades D3DX, DXGI y las bibliotecas de versiones antiguas de Direct3D. Estos componentes simplifican diferentes tareas en la programación gráfica.
- ✓ El componente DirectInput para soporte de gran variedad de dispositivos de entrada incluyendo soporte para la tecnología *force-feedback*.
- ✓ El *Windows Games Explorer* que provee una forma personalizada para presentar los videojuegos a los usuarios de Microsoft *Windows*.

El SDK también contiene un conjunto de artículos técnicos escritos por expertos en la tecnología que de forma práctica ayudan al aprendizaje del uso de estos componentes. Estos recursos son:

- ✓ DirectX *Developer Center:* contiene la última información sobre el API y las versiones más recientes para ser descargadas. Enlace: http://msdn.microsoft.com/en-us/directx/default.aspx

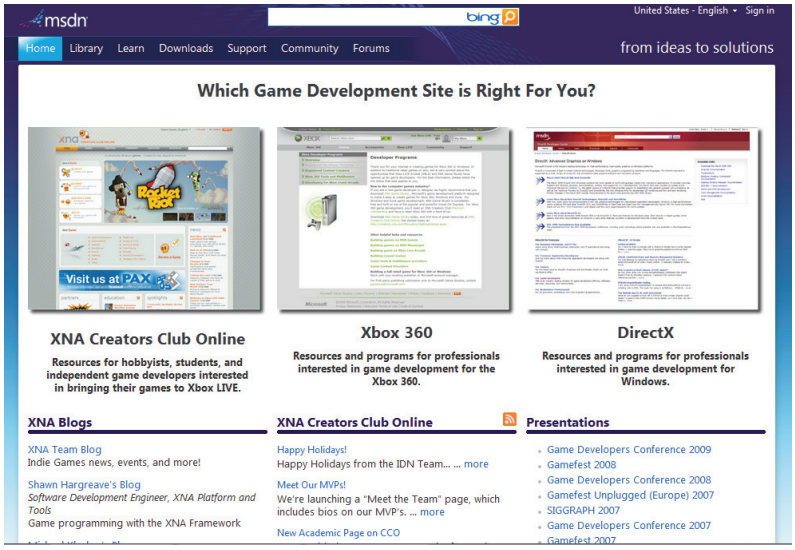
Figura 27. **Microsoft DirectX *Developer***



Fuente: DirectX Developer Center. http://msdn.microsoft.com/en-us/directx/default.aspx.

✓ XNA *Game Studio Express*: formado por un conjunto de herramientas y tecnologías orientadas al apoyo de estudiantes y programadores aficionadas para las plataformas *Windows* y Xbox 360.

Figura 28. **Microsoft XNA *Game Developer***



Fuente: Microsoft XNA Game Developer. http://msdn.microsoft.com/en-us/directx/default.aspx.

- ✓ GDI+: es una tecnología de *software* diseñada para la creación de textos de alta calidad. Es un API basado en clases para programadores en lenguaje C y C++.

1.10.1.2. OpenGL

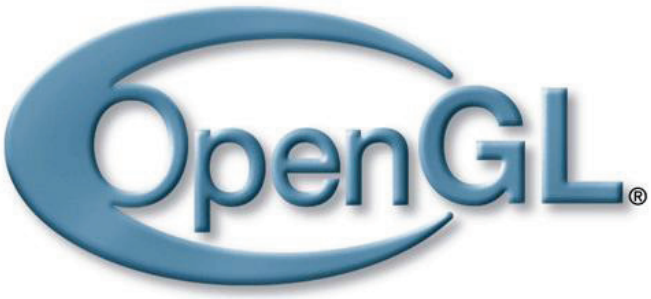
OpenGL es una especificación estándar desarrollada originalmente por *Silicon Graphics Inc*. (SGI) que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. Existen implementaciones de OpenGL para Microsoft *Windows*, Linux, Mac OS, Unix y *PlayStation 3*. También existen diversas implementaciones de *software* que permiten la ejecución de aplicaciones que dependen de OpenGL sin el soporte de aceleración por *hardware* y la variante simplificada OpenGL ES (OpenGL *for Embedded Systems*) diseñada para dispositivos integrados como teléfonos móviles, PDAs y otras consolas de videojuegos.

OpenGL provee al programador de una interfaz al *hardware* de video. Es una biblioteca poderosa de renderizado de bajo nivel disponible para la mayoría de plataformas con un amplio soporte de *hardware*. Está diseñado para ser usado en cualquier aplicación gráfica, desde videojuegos hasta modeladores CAD. Muchos juegos, como *Doom* 3 de Id *Software* usan OpenGL para su núcleo de renderizado gráfico (Astle, 2004).

La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples. Principalmente es una especificación que describe un conjunto de funciones y el comportamiento que éstas deben tener. A partir de estas especificaciones, los fabricantes de *hardware* crean las implementaciones y bibliotecas de funciones que se deben ajustar a los requisitos de la especificación y superar los test de conformidad de OpenGL para así poder utilizar el logotipo oficial de OpenGL.

OpenGL ES ha sido seleccionada como la API para gráficos 3D oficial en el sistema operativo Symbian OS y la plataforma para dispositivos móviles *Android*. También es la biblioteca gráfica 3D en el SDK del iPhone.

Figura 29. **Logo oficial de OpenGL**



Fuente: SGI. http://appleweblog.com/2010/01/soporte-de-opengl-30-en-mac-os-x-1063.

El diseño de OpenGL tiene dos propósitos esenciales:

- ✓ Ocultar la complejidad de la interfaz con las diferentes tarjetas gráficas, presentando al programador una API única y uniforme.
- ✓ Ocultar las diferentes capacidades de las diversas plataformas *hardware*, requiriendo que todas las implementaciones soporten la funcionalidad completa de OpenGL.

1.10.2. Otras herramientas

Además de los lenguajes y bibliotecas existen herramientas que proveen altos niveles de funcionalidad y características como animación de esqueletos, simulación física, soporte de modelos en 3D y otras complejas tecnologías como la conversión del programa ejecutable a una plataforma distinta. Otras herramientas permiten el control del código fuente, optimización, depuración facilitando la escritura del código y detección de problemas.

La mayoría de programadores profesionales utiliza un IDE (*Integrated Development Environment*) que contiene un editor de código fuente, un diseñador de interfaz gráfica, un compilador y un depurador. De igual forma, la selección del IDE dependerá de la plataforma de destino. Un IDE muy popular para el desarrollo de programas para Microsoft *Windows* y Xbox es Microsoft *Visual Studio* .NET.

Muchas de las grandes empresas de desarrollo crean sus propias herramientas personalizadas para ser utilizadas exclusivamente por ellas como editores de niveles para el videojuego.

1.10.3. Microsoft XNA

Microsoft XNA (XNA *is Not an Acronym*) es un conjunto de herramientas con un entorno de ejecución proporcionado por Microsoft que facilita el desarrollo de videojuegos.

Figura 30. **Microsoft XNA**

Fuente: Microsoft Corporation. http://create.msdn.com/en-US/.

Busca liberar a los programadores de juegos de la creación de código repetitivo e incluye un conjunto diferentes herramientas para la producción de videojuegos. Abarca secciones de Microsoft *Game Development Sections*, incluyendo el estándar *kit* de desarrollo de Xbox y XNA *Game Studio*.

1.10.3.1. XNA *Framework*

XNA *Framework* se basa en la implementación nativa de .NET *Compact Framework* 2.0 para el desarrollo de la Xbox 360 y .NET *Framework* 2.0 en *Windows*.

Consta de un conjunto de bibliotecas de clases específicas para el desarrollo de juegos que permiten la reutilización de código a través de plataformas de destino.

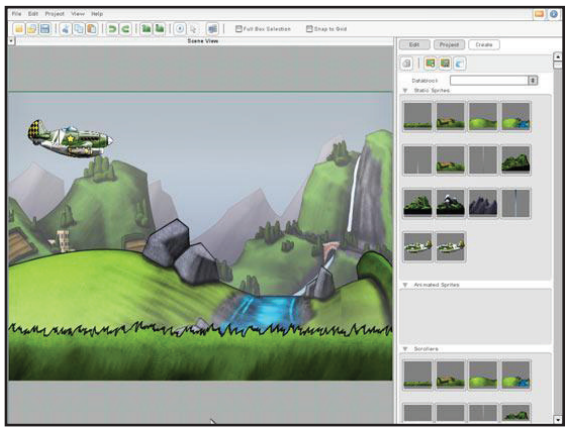
1.10.3.2. XNA *Game Studio*

Es un entorno de desarrollo integrado (IDE) para el desarrollo de videojuegos. Las versiones existentes son:

- ✓ XNA *Game Studio Professional*: versión dirigida a desarrolladores profesionales de videojuegos. Proporciona una estructura colaborativa entre los distintos elementos del grupo de desarrollo y herramientas para gestión y administración del proyecto.

- ✓ XNA *Game Studio Express*: versión disponible de forma gratuita destinada a estudiantes, aficionados y creadores independientes. Permite la creación de videojuegos para Microsoft *Windows* sin ningún costo con el XNA *Framework* y pagada para Xbox 360.

Figura 31. **Microsoft XNA *Game Studio***



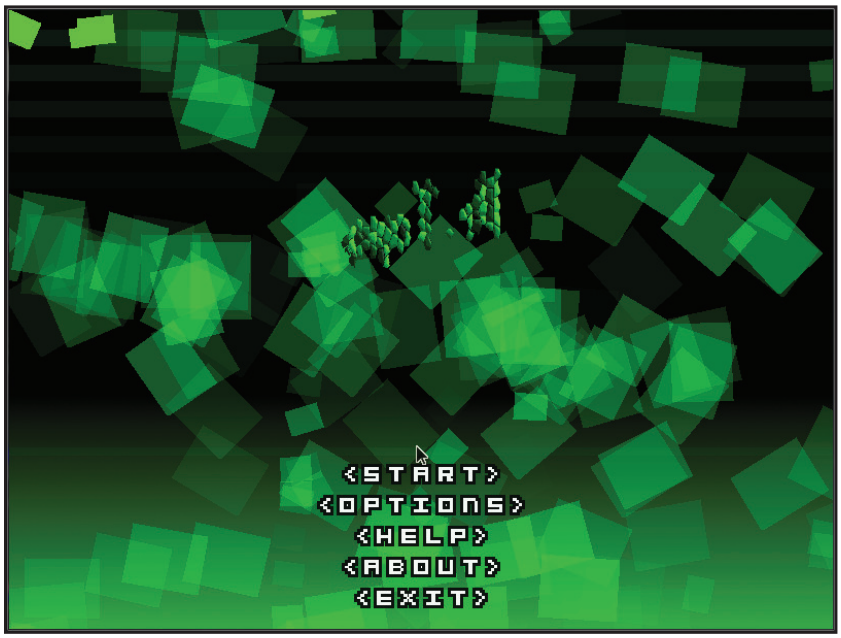
Fuente: Xbox666. http://xbox666.com/category/xbox-360/aplicaciones/page/3/.

2. ANÁLISIS Y SELECIÓN DE LAS HERRAMIENTAS PARA LA IMPLEMENTACIÓN DEL VIDEOJUEGO

2.1. Breve descripción

El videojuego a desarrollar consiste en un juego del género de plataformas con vista de *scroll* lateral que ejemplifica el proceso de desarrollo de videojuegos independiente. Este se implementará en lenguaje de programación Div utilizando los compiladores BennuGD y Gemix.

Figura 32. **Dr. Topo**



Fuente: propia.

2.2. Selección del lenguaje de programación

2.2.1. Lenguaje de programación

- ✓ DIV

La selección del lenguaje se ha realizado en base a los siguientes criterios:

- ✓ DIV es un lenguaje de programación orientado a procesos. Esto facilita la programación del videojuego ya que cada elemento se comporta como un proceso independiente.
- ✓ DIV es un lenguaje familiar y conocido.
- ✓ El compilador Gemix permite la generación de ejecutables multiplataforma (*Windows*, Linux y MacOS a partir de la versión 6.0).

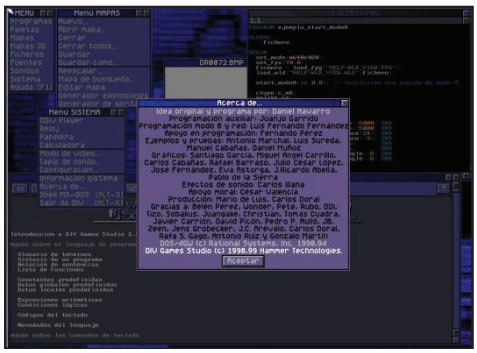
2.3. Historia del lenguaje

DIV es un lenguaje de programación que tiene una estructura basada enteramente en el lenguaje de programación C. Fue creado por Daniel Navarro y se desarrollo en un principio para la creación de juegos en el entorno de MS-DOS. La empresa *Hammer Technologies* lo comercializó como Div *Games Studio* 1 y Div *Games Studio 2*.

En los años 90, Daniel Navarro comenzó a desarrollar una herramienta para la creación de videojuegos que luego se convertiría en el lenguaje DIV. Este lenguaje combinaba características de los lenguajes C y PASCAL en un entorno complejo que nació con la intención de ser un lenguaje sencillo y fácil de aprender.

Div *Games Studio* incluía un compilador, un programa de diseño de *sprites* y un programa de edición de archivos de sonido, y otras herramientas que permitían la creación completa de un videojuego.

Figura 33. **Div *Games Studio* 2 de *Hammer Technologies***



Fuente: propia.

Con la aparición del sistema operativo Microsoft *Windows* XP, que no era compatible con los programas compilados con DIV2 y la quiebra de la empresa distribuidora, el lenguaje como producto comercial quedo abandonado.

El principal culpable de la muerte de *Hammer* no fueron los miembros que la integraban, sino los que hicieron realidad el proyecto como empresa. Su total incompetencia, mala gestión y desprecio por el trabajo que sus propios empleados estaban realizando en *Hammer*, terminó por desembocar, primero, en una estampida de las personas más relevantes de *Hammer* a empresas como *Dinamic*, Pyro o incluso al extranjero (casos como el de César Valencia, Jorge Rosado, Daniel Navarro o David Picón) y, segundo, en una progresiva pérdida de motivación en los empleados que aún quedaron y un desprestigio que se fue acentuando según avanzaba el tiempo. Fue una lástima y sobre todo, un insulto a la confianza e ilusión que mucha gente depositó en esta empresa. Esperamos que ciertos personajes no vuelvan a aparecer nunca más en este sector (Rodríguez, 2001).

2.4. Características del lenguaje

DIV es un lenguaje de programación orientado a procesos. Un lenguaje de programación orientado a procesos es altamente recomendado para la creación de videojuegos debido a que permite programar cada elemento de forma independiente. Cada vez que un programa llama a un procedimiento, se crea una copia de éste como proceso. El proceso se ejecuta una vez durante cada fotograma o *frame* del juego.

2.5. Selección de compiladores

2.5.1. BennuGD

BennuGD nace como un *fork* del proyecto Fénix y se distribuye bajo licencia GNU *General Public License*. Es una herramienta simple y muy completa, que gracias a la posibilidad de importar bibliotecas de enlace dinámico programadas en C y C++ prácticamente no tiene límites.

Fénix es un proyecto multiplataforma de *software* libre elaborado por José Luis Cebrián para la creación de un compilador alternativo al lenguaje DIV. Desde la versión 0.71 el proyecto quedó estancado, lo que dio lugar a múltiples versiones derivadas que corregían fallos o añadían nuevas características. A partir del año 2006, el proyecto ha sido continuado por SplinterGU, el mismo que implementó el primer sistema de bibliotecas de enlace dinámico para el compilador y quien luego decidió crear un *fork* nombrado BennuGD (Bennu *Game Development*).

Figura 34. **Bennu *Game Development***

Fuente: http://www.bennugd.org/.

BennuGD se enfoca en la modularidad y portabilidad haciéndolo una gran opción para el desarrollo multiplataforma de videojuegos. BennuGD presenta una gran cantidad de *bugs* corregidos respecto a Fénix además de una serie de importantes innovaciones. A pesar de esto, se mantiene la compatibilidad entre ambos lenguajes.

BennuGD cuenta con un fichero intermedio entre el entorno de compilación y el entorno de ejecución por lo que no es necesario distribuir el código fuente de un juego para poder utilizarlo.

Oficialmente soporta *Windows*, Linux y GP2X Wiz aunque existen versiones no oficiales que pueden soportar BSD, MacOSX y las consolas Wii, Dingoo A320, GP2X y Xbox.

La posibilidad de utilizar extensiones permite ampliar el lenguaje en aspectos como manejo de red, manipulación avanzada de gráficos, reproducción de distintos formatos de audio y video, XML *parsing*, renderizado, etc. Por ejemplo es posible reemplazar el renderizado por *software* predeterminado con el *render* 3D OpenGL basado en el *engine* Irrlicht o utilizar los mapas BSP y modelos del *engine* de *Quake* 3 de Id *Software*.

Figura 35. **Extensión Bennu3D**

Fuente: http://www.bennugd.org/.

2.5.1.1. Características

- ✓ Es un lenguaje interpretado;
- ✓ Los programas compilados pueden ser intercambiados entre plataformas sin necesidad de recompilación;
- ✓ Altamente portable, modular y multiplataforma;
- ✓ Sistemas Operativos soportados:
  - o *Windows* (9x, ME, 2000, XP x86/x64, Vista x86/x64, 7 x86/x64);
  - o Linux (x86, x64, ARM, PPC);
- ✓ Consolas (Wiz., Xbox, Wii, GP2X);
- ✓ Procesos mediante programación multihilo;
- ✓ Motor 2D de renderizado por *software*;
- ✓ Soporte y manejo de expresiones regulares;
- ✓ Modos gráficos 8, 16 y 32 *bits* con soporte de rotación de *sprites*, escalado, manejo de *flags*, *alpha blending,* blendops, blit aditivo y sustractivo, etc.;
- ✓ Soporte para reproducción de sonidos en formato wav, pcm y ogg vorbis;
- ✓ Soporte de bibliotecas DLL en las plataformas compatibles.

2.5.2. Gemix *Studio*

Gemix *Studio* es una herramienta para el desarrollo de videojuegos que tienen como objetivo incorporar un IDE con las herramientas necesarias para la creación de videojuegos y ser totalmente compatible con el lenguaje de programación Div.

Figura 36. **Gemix *Studio***

Fuente: http://www.gemixstudio.com/.

Esta herramienta actualmente se encuentra en desarrollo y comparte la filosofía original de Div *Games Studio*: la creación de un entorno de edición y una serie de lenguajes que faciliten considerablemente la creación de videojuegos tanto a programadores novatos como experimentados. Gemix *Studio* está compuesto fundamentalmente por cuatro secciones:

- ✓ Un IDE que integra el máximo número de herramientas que permitan la edición de los diferentes recursos que toman parte en un videojuego (código fuente, gráficos, bibliotecas de gráficos, sonidos, fuentes, música etc.) y un variado conjunto de utilidades que simplifiquen el proceso.
- ✓ Una serie de lenguajes y compiladores.
- ✓ Un intérprete que ejecuta el código intermedio producido por los compiladores.
- ✓ Una serie de módulos que aportan diversas funcionalidades.

2.5.2.1. Características

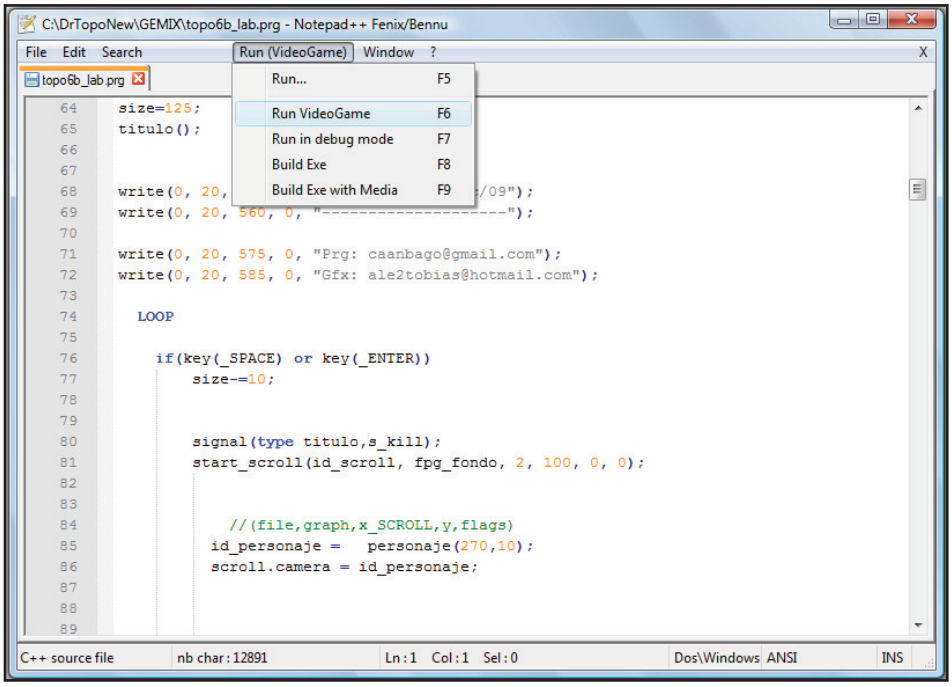
- ✓ Es un lenguaje interpretado. El compilador de Gemix convierte el código fuente en código intermedio que posteriormente es ejecutado por el intérprete;
- ✓ Multiplataforma: *Windows*, Linux y Mac (a partir de la versión 6.0);
- ✓ Motor 2D de renderizado por *software* con sistema de efectos y modos de fusión de gráficos;
- ✓ Modos gráficos de 8, 16 y 32 *bits*;
- ✓ Motor de *scroll parallax* automático de dos planos;
- ✓ Motor de modo 7;
- ✓ Cuenta con una amplia colección de funciones para la manipulación de cadenas de texto;
- ✓ Soporte para reproducción de sonidos en formato wav, pcm y ogg vorbis, mp3, etc. y diversas funcionalidades para la manipulación en tiempo real del audio;
- ✓ Reproducción de módulos de música en formato it, s3m, mod, xm y midi.

2.6. Otras herramientas utilizadas

2.6.1. *Noteptad++ for* FENIX/BENNU V5.03

Versión modificada del editor de texto de código libre de Don Ho, compatible con la sintaxis del lenguaje Div y configurado para la compilación y ejecución del código fuente incluido en el paquete devBennu.

Figura 37. ***Notepad++ for* Fenix/Bennu**

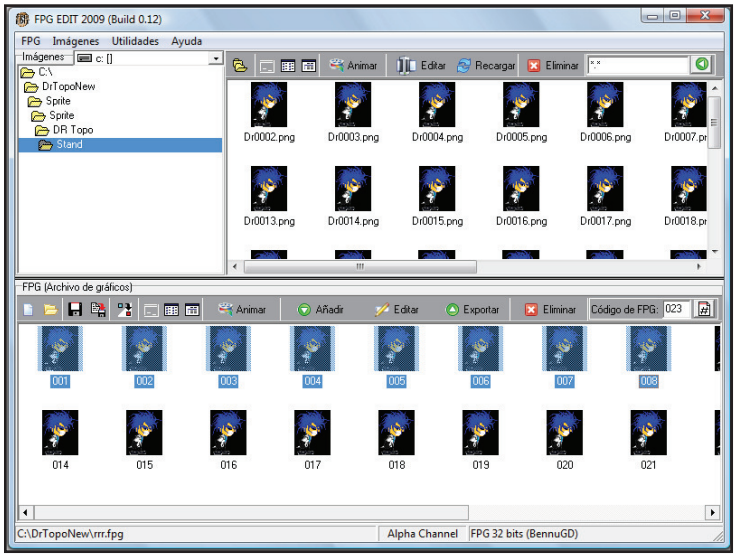


Fuente: propia.

2.6.2. FPG *Edit* V0.12

FPG *Edit* es una herramienta de código libre que permite editar y visualizar archivos de imágenes FPG para aplicaciones realizados en DIV2, CDIV, Fénix, BennuGD y Gemix.

Figura 38. **FPG *Edit***



Fuente: propia.

2.6.2.1. Características

- ✓ Soporte de archivos FPG:
  - FPG de 1 *bit*
  - FPG de 8 *bits* para (DIV2, CDIV, Fénix, BennuGD y Gemix)
  - FPG de 16 *bits* para Fénix y BennuGD
  - FPG de 16 *bits* para CDIV
  - FPG de 24 *bits*
  - FPG de 32 *bits* para BennuGD

- ✓ Formatos de imágenes soportadas:
  - PNG, MNG, JNG, JPG, JIF, GIF, BMP, WMF, EMF, DIB, RLE, TGA, PCX, ICO, TIF, TIFF, FAX, BW, RGB, RGBA, SGI, CEL, PIC, VST, ICB, VDA, WIN, PCC, SCR, PCD, PPM, PGM, CUT, PAL, RLA, RPF, PSD, PDD, PSP

- ✓ Formatos de imágenes soportadas:
  - PAL

- ✓ Importe de paletas de ficheros de 8 *bits*:
  - BMP, PCX, FPG, MAP y FNT

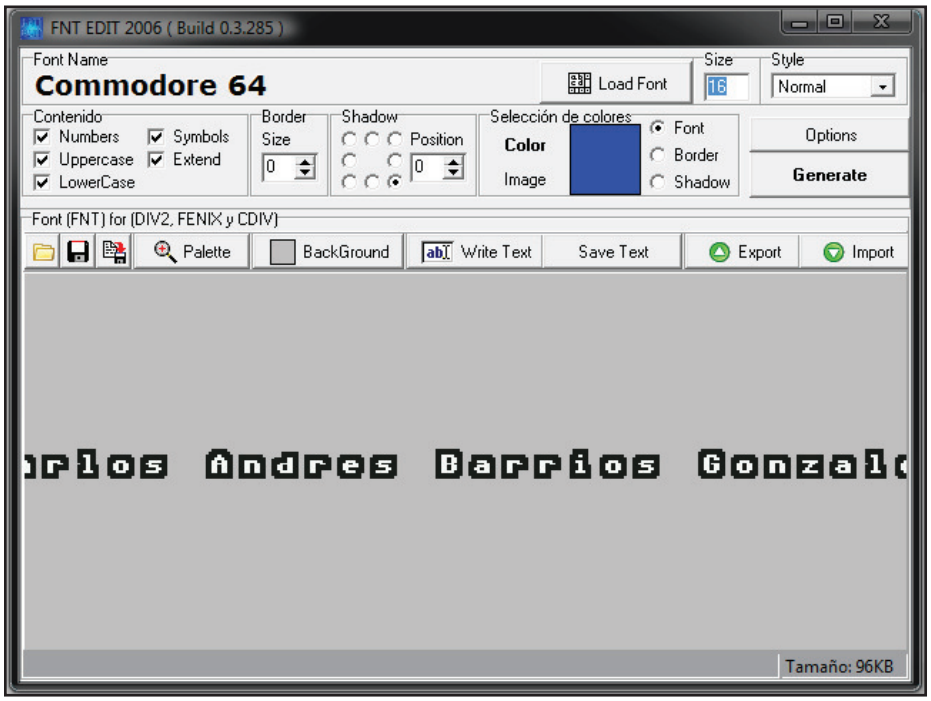
- ✓ Importe de paletas de ficheros de 8 *bits*:
  - CPT

- ✓ Conversión entre formatos;
- ✓ Sistema de multilenguaje autoadaptable;
- ✓ Manejo del portapapeles para imágenes;
- ✓ Sistema de animación para las imágenes de los ficheros FPG;
- ✓ Filtrado de imágenes;
- ✓ Compresor para FPG de Fénix/BennuGD.

2.6.3. FNT *Edit* V0.3.285

FNT *Edit* es una herramienta que permite la conversión de fuente TTF a formato FNT compatible con Div2, Fénix, BennuGD y Gemix.

Figura 39. **FNT *Edit***

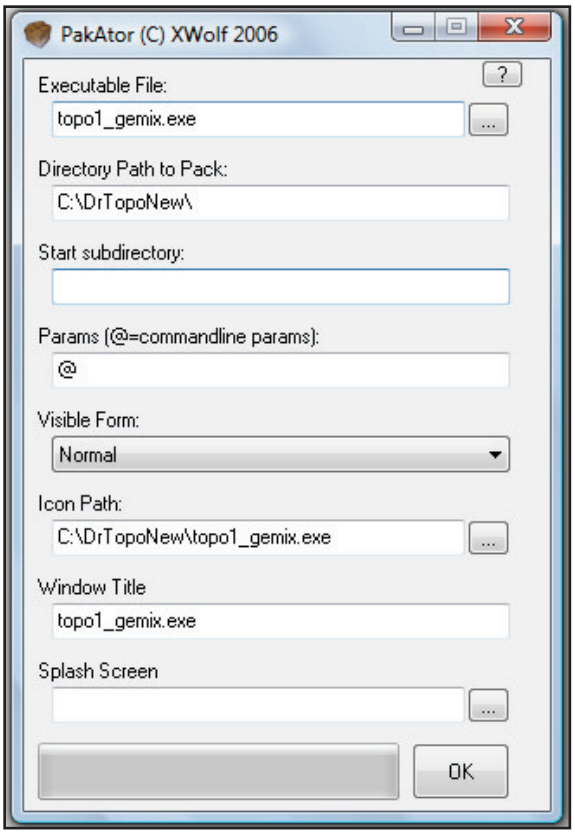


Fuente: propia.

2.6.4. PakAtor

Herramienta creada por Iván Domínguez que permite empaquetar todos los archivos del videojuego (gráficas, archivos FPG, fuentes, sonidos, etc.) en un solo archivo ejecutable.

Figura 40. **PakAtor**

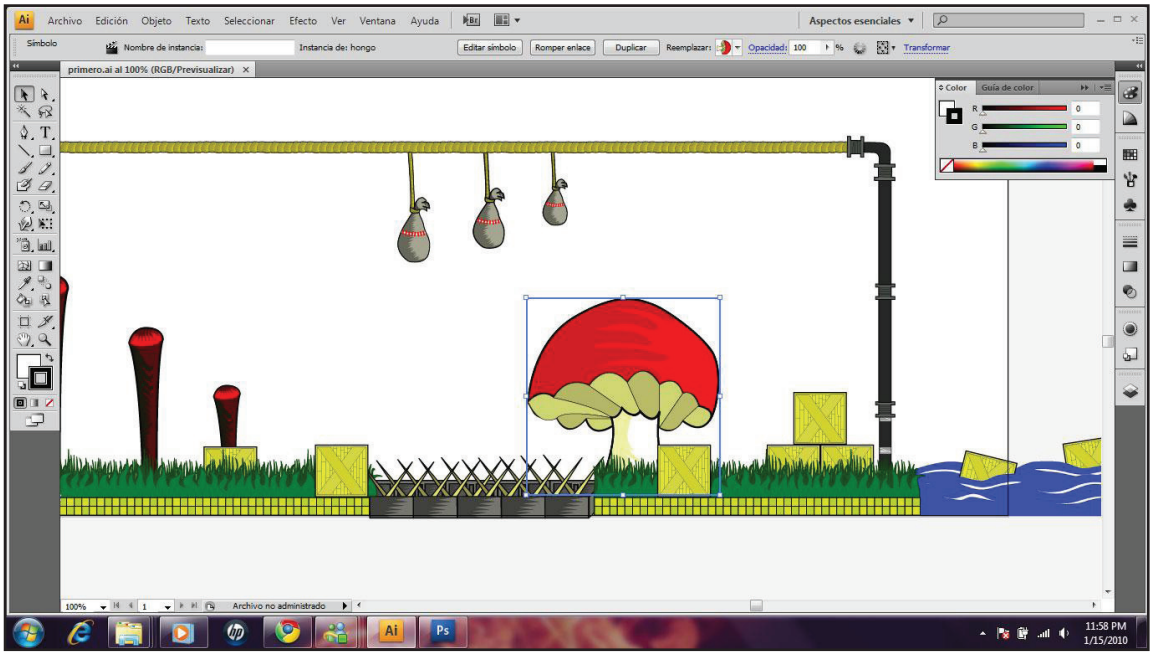


Fuente: propia.

2.6.5. *Adobe® Illustrator® CS4*

Software utilizado para la creación de *sprites*, imágenes, fondos, etc. del videojuego. *Adobe Illustrator CS4* es un entorno gráfico de vectores que incluye transparencias de degradados y mesas de trabajo que facilitan la etapa de diseño y creación del arte digital.

Figura 41. **Ejemplos de las gráficas creadas para el videojuego Dr. Topo**

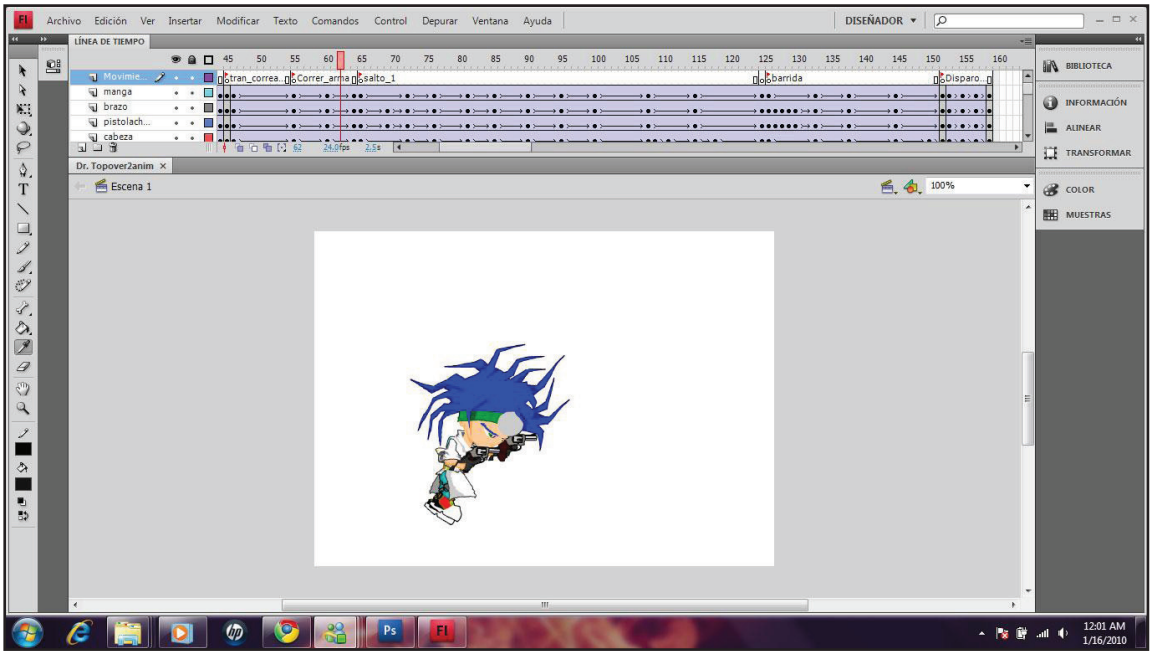


Fuente: propia.

2.6.6. *Adobe® Flash® CS4*

Herramienta que trabaja con fotogramas, gráficos vectoriales e imágenes *raster* utilizada para la animación de los personajes del videojuego. Al permitir la manipulación de gráficos vectoriales facilita la creación de los *sprites* de los movimientos del personaje principal y de los enemigos del videojuego.

Figura 42. **Animación en *Adobe Ilustrator CS4* utilizada para los *sprites***

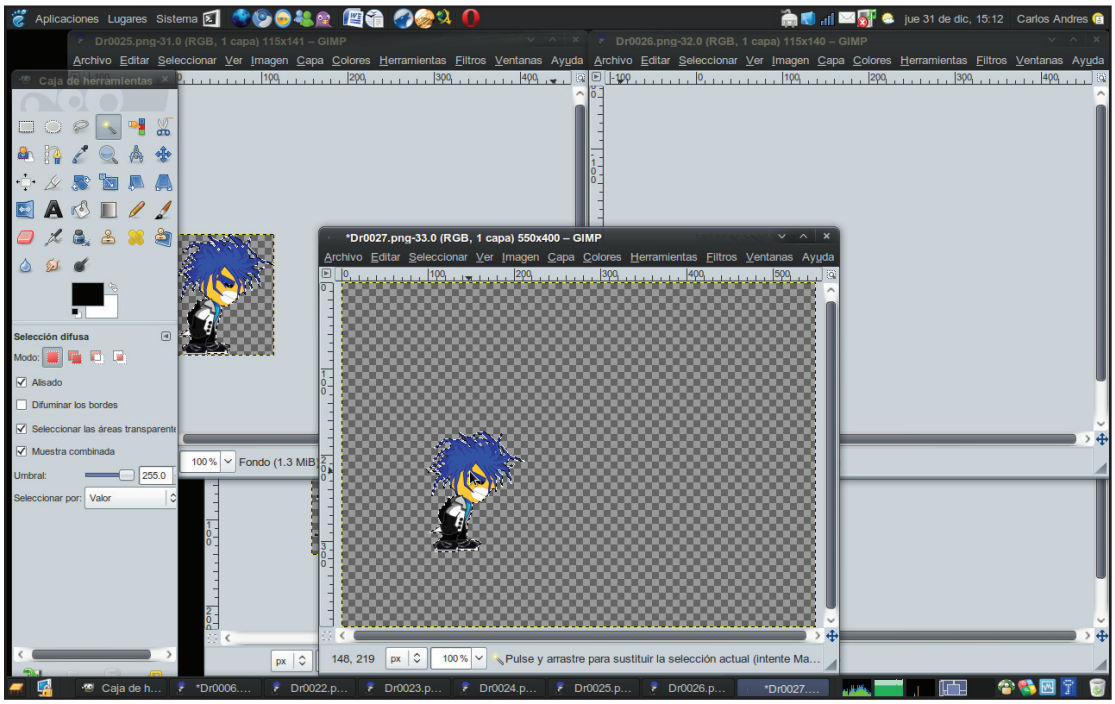


Fuente: propia.

2.6.7. GIMP

GIMP (GNU *Image Manipulation Program*) es un programa de edición de imágenes digitales en forma de mapa de *bits* disponible bajo licencia GNU siendo libre y gratuito. Esta herramienta fue utilizada para la edición y retoque de los *sprites* generados para las animaciones de los personajes e elementos dinámicos del videojuego.

Figura 43. **Edición de *sprites* utilizando GIMP**

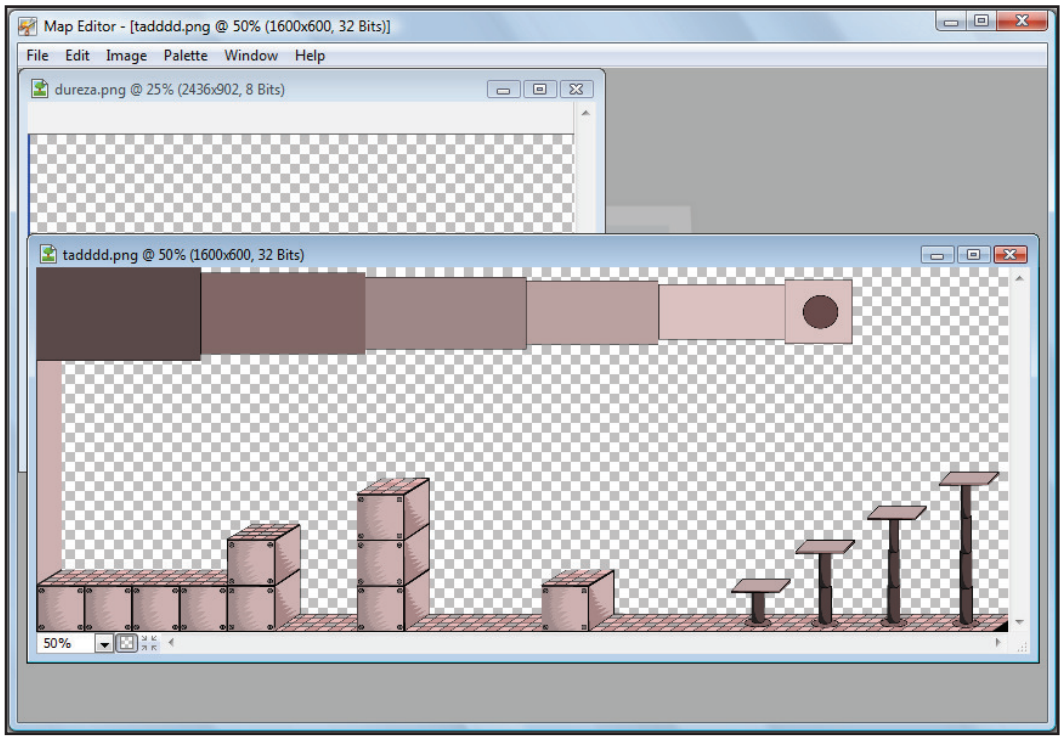


Fuente: propia.

2.6.8. *Map Editor*

Herramienta de edición de imágenes de SkyGem *Software* utilizada para la conversión de imágenes a formato MAP para ser cargadas a los archivos gráficos FPG.

Figura 44. ***Map Editor* de SkyGem *Software***

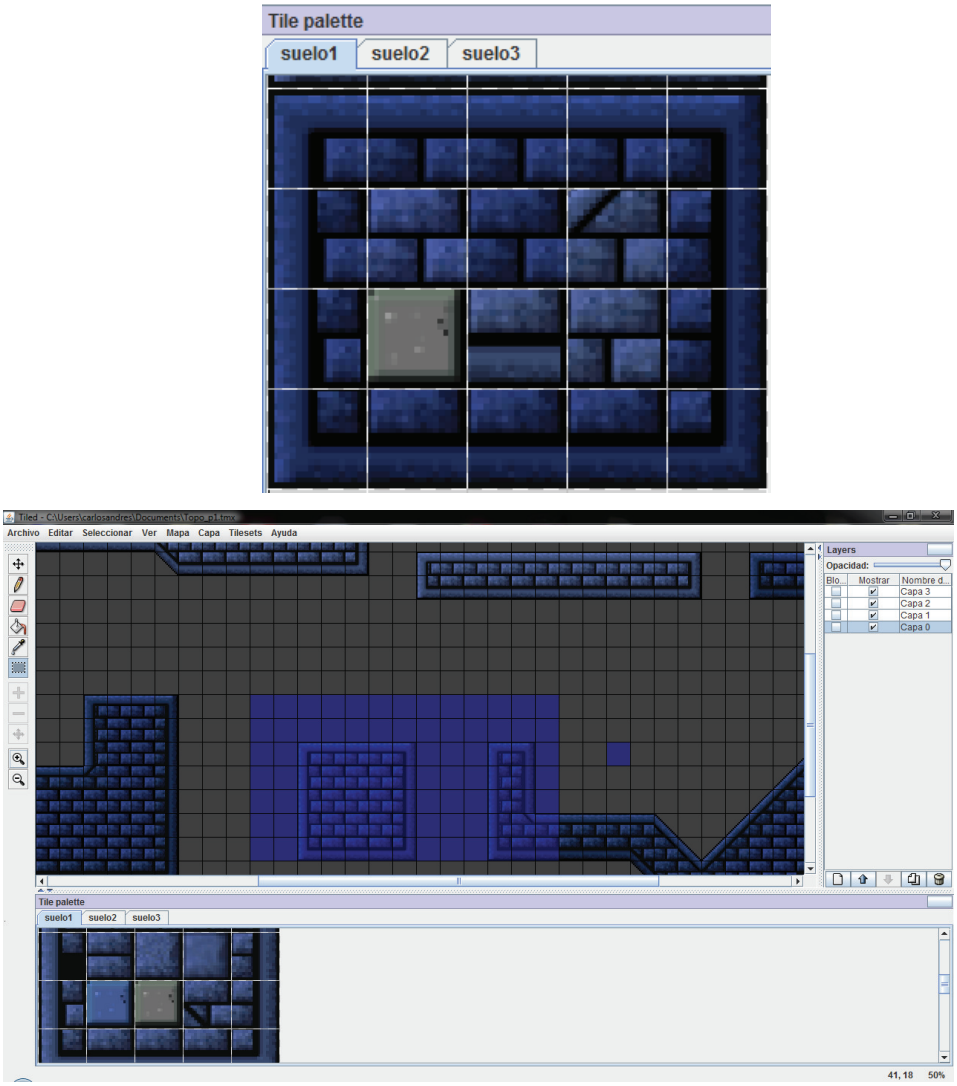


Fuente: propia.

2.6.9. *Tiled Map Editor*

Herramienta para edición de mapas a partir de *tiles* o baldosas que puede trabajar con mapas ortogonales e isométricos basándose en el formato XML con flexibilidad para trabajar con distintos *game engines*.

Figura 45. **Edición de mapas para el videojuego en *Tiled Map Editor***

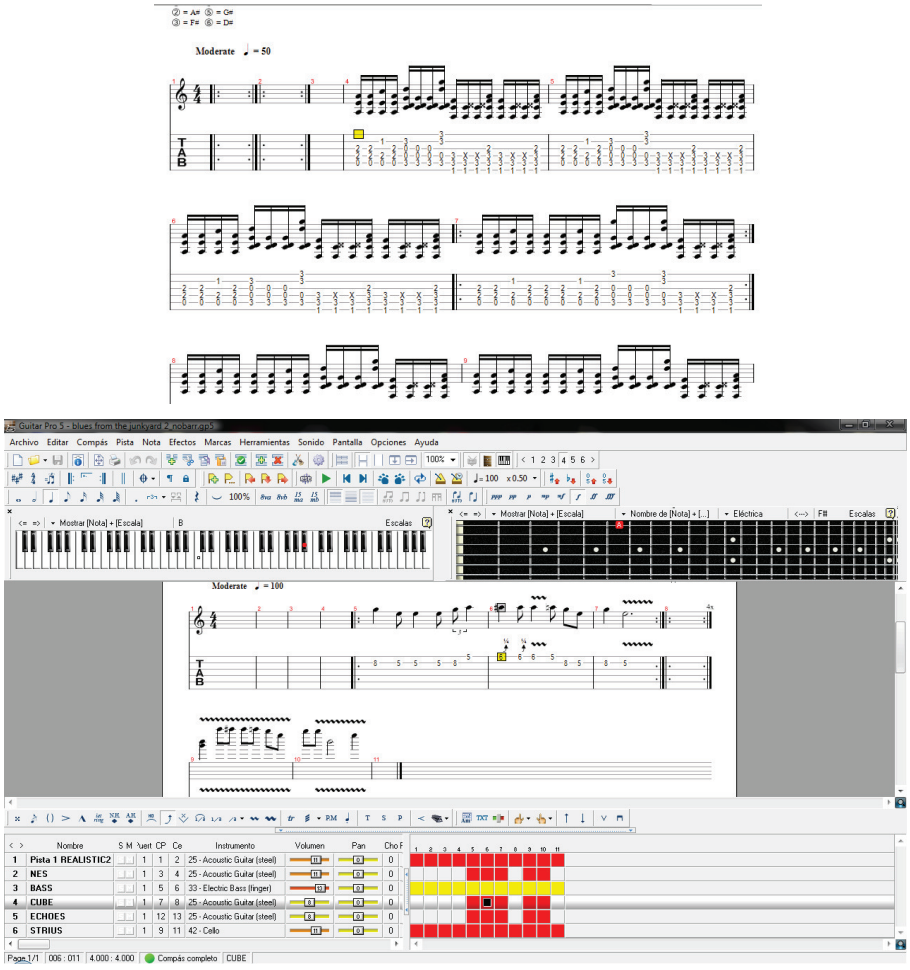


Fuente: propia.

2.6.10. *Guitar Pro*

Programa para edición de tablaturas y partituras especialmente para guitarra desarrollado por Arobas *Music*. *Guitar Pro* soporta el formato e interfaz MIDI. A partir de la versión 5.0 incorpora RSE (*Realistic Sound Engine*), una herramienta que reproduce sonidos pregrabados con un sonido realista en lugar de la reproducción MIDI.

Figura 46. **Edición de partituras en *Guitar Pro* para *Windows***

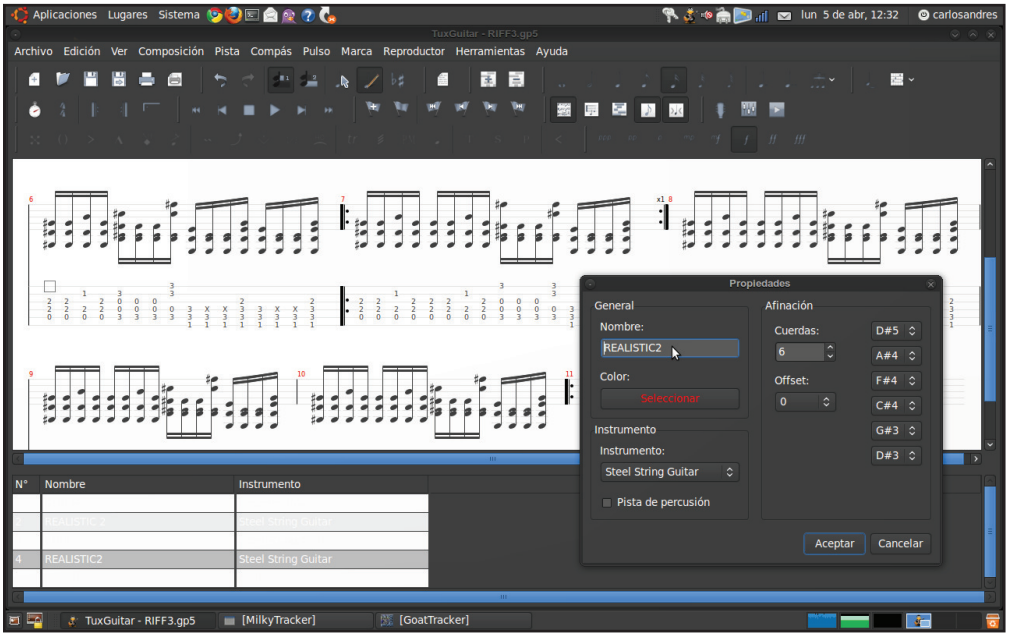


Fuente: propia.

2.6.11. *TuxGuitar*

Editor de tablaturas y partituras multiplataforma compatible con los sistemas operativos Linux, Mac OS y *Windows* de licencia libre desarrollado por Herac. *TuxGuitar* es compatible con los formatos de *Guitar Pro* y *PowerTab* que permite la importación y exportación de archivos MIDI.

Figura 47. **Edición de partituras en *TuxGuitar* para Linux**



Fuente: propia.

2.6.12. FL *Studio* 9

Estación de audio digital desarrollado por *Image-line Software* que permite la utilización de *samples*, generadores por *software* (VSTi, DXi y softsynth) e instrumentos MIDI para la edición y producción de música digital.

Cuentan con un secuenciador basado en patrones, sintetizadores digitales, bibliotecas de *samples*, mezcladores y gran cantidad de efectos utilizando *plugins* compatibles con los estándares VST, VST2, Buzz, DirectX y Rewire soportando los *drivers* ASIO y DirectSound.

Figura 48. **Entorno de trabajo de FL *Studio* 9**

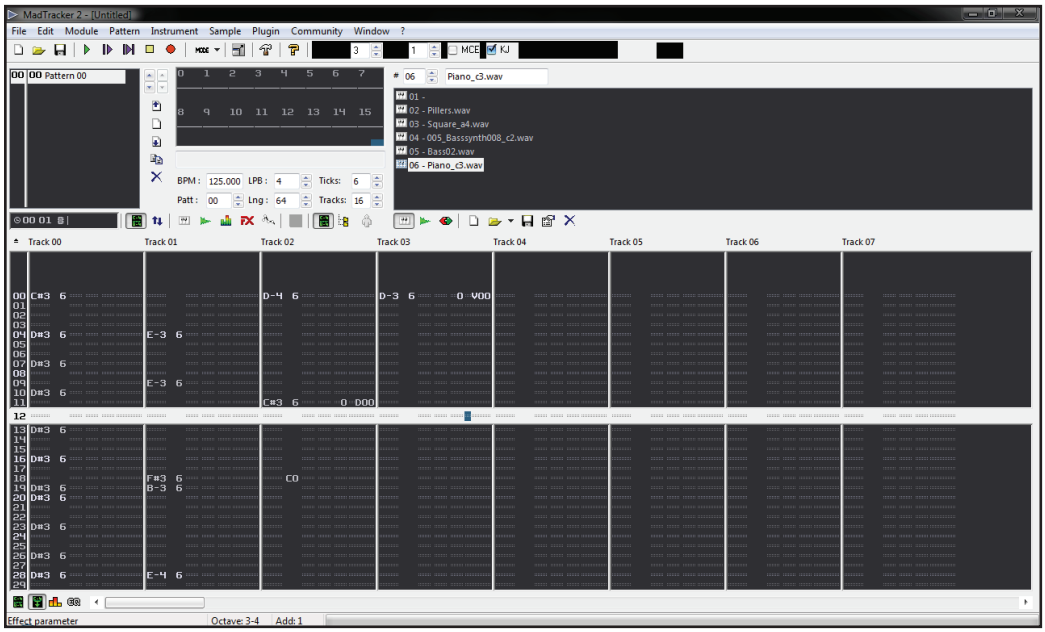
Fuente: propia.

2.6.13. *MadTracker 2*

MadTacker es una herramienta de composición basada en el concepto de *tracker* compatible con los estándares VST, Asio y Rewire.

Un *tracker* es una herramienta de *software* que funciona como secuenciador utilizando *samples* o muestras digitales en distintos canales. Las notas musicales son generalmente representadas por caracteres alfanuméricos y códigos hexadecimales.

Figura 49. **Edición de *tracks* en *MadTracker 2***



Fuente: propia.

2.7. Resumen

2.7.1. Compiladores

Tabla II. **Versiones de los compiladores utilizados para el videojuego**

NOMBRE	VERSIÓN	TIPO DE LICENCIA
BennuGD	1.0.0	GNU GPL
Gemix	0.5.7 (Beta) Pública 0.6.0 (Beta) RC7 de uso interno.	PROPIETARIA

Fuente: propia.

2.7.2. Herramientas

Tabla III. **Herramientas utilizadas para la creación del videojuego**

NOMBRE	VERSIÓN
NOTEPTAD++ FOR FENIX/BENNU	5.3
FPG *EDIT*	0.12
FNT *EDIT*	0.3.285
PAKATOR	0.1
ADOBE® ILLUSTRATOR® CS4	14.0.0
ADOBE® FLASH® CS4	10.0.2

Continúa tabla III

GIMP	2.6.8
MAP EDITOR	0.3
TILED MAP EDITOR	0.7.2
GUITAR PRO	5.2
TUXGUITAR	1.2
FL *STUDIO* 9	9.0.3
MADTRACKER 2	2.6.1

Fuente: propia.

3. PROCESO DE DESARROLLO

El proceso de desarrollo del videojuego sigue las directivas expuestas por Clinton Keith en el libro "*Agile Game Development with Scrum*" como modelo de referencia para ajustar la filosofía de *Scrum*, sus procesos, fases y elementos a las características únicas del desarrollo de videojuegos.

Su enfoque iterativo e incremental junto a los principios básicos del desarrollo de *software* ágil lo convierten en una de las mejores metodologías adaptables al desarrollo de videojuegos independientes.

3.1. Fase de Pre-Producción

Durante esta fase del proceso de desarrollo del videojuego se definieron los siguientes aspectos fundamentales para su creación:

3.1.1. Género

- ✓ Plataformas con desplazamiento lateral (*scroll*).

3.1.2. *Game play*

Dr. Topo sigue la temática de juego de un videojuego de *scroll* en la que el protagonista que debe de recorrer y saltar a través de una serie de plataformas, obstáculos y enemigos para avanzar a la siguiente fase. Al avanzar por los niveles del videojuego, debe recolectar una seria de objetos y premios que modifican las cualidades y capacidades del protagonista.

Si el protagonista logra completar el nivel, este avanza a la siguiente fase hasta enfrentarse al enemigo final.

El nivel está rodeado por elementos sólidos que no pueden ser traspasados por el personaje, formando las durezas del nivel, las plataformas, paredes, techos y otros elementos que constituirán el universo del videojuego.

Figura 50. **Prueba del primer nivel del videojuego**

Fuente: propia.

3.1.3. Mecánica

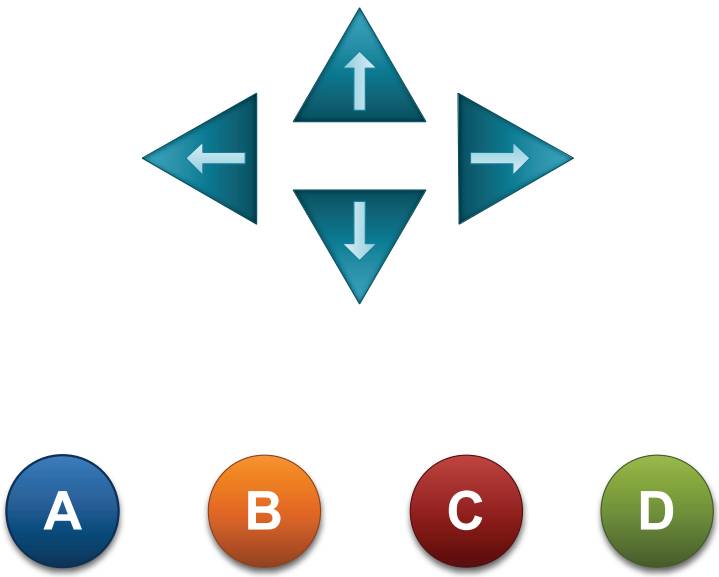
Las interacciones del usuario con el videojuego se realizarán mediante el teclado, *joypad* y *mouse*. Los elementos de los menús e interfaces del videojuego se controlan mediante el *mouse*. Los movimientos del personaje principal pueden realizarse mediante el teclado o un *joypad*.

Las acciones definidas para la interacción con el personaje principal son:

- ✓ Acción caminar a la derecha
- ✓ Acción caminar a la izquierda
- ✓ Acción saltar
- ✓ Acción golpe#1
- ✓ Acción golpe#2
- ✓ Acción salto/giro
- ✓ Acción disparo
- ✓ Sin acción (ninguna tecla presionada)

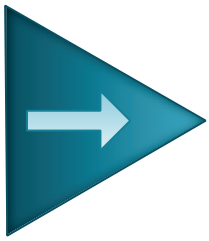
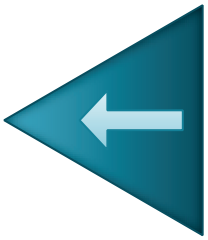
3.1.4. Controles y acciones

Figura 51. **Controles y acciones para el personaje principal**

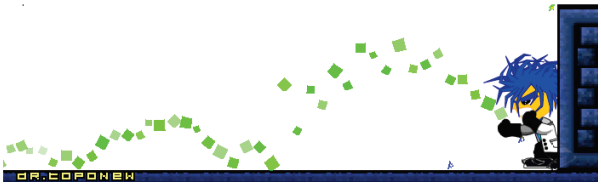


Fuente: propia.

Tabla IV. **Acciones básicas del personaje principal**

TECLA	ACCIÓN	EJEMPLO
	Saltar	
	Correr hacia el lado derecho	
	Correr hacia el lado izquierdo	

Continúa tabla IV

A	Golpe	
B	Patada	
C	Salto con giro	
D	Disparo de bloques	

3.1.5. Reglas del ambiente

Cada nivel que constituirá el mundo virtual del videojuego estará formado por límites sólidos básicos que son:

Tabla V. **Colores de durezas**

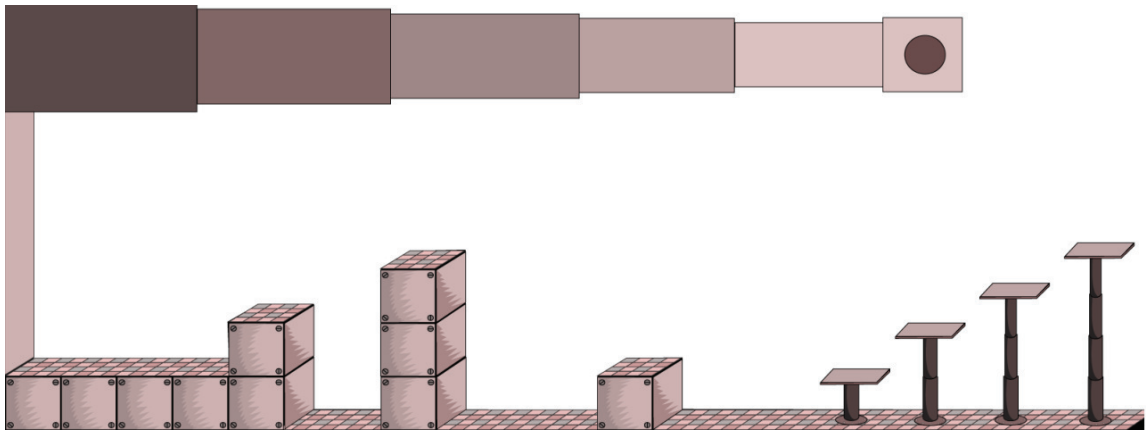
REGIÓN	COLOR	RGB
Suelo		(255,0,0)
Pared		(0,0,255)
Techo		(0,255,0)
Otros sólidos		(255,255,255)

Fuente: propia.

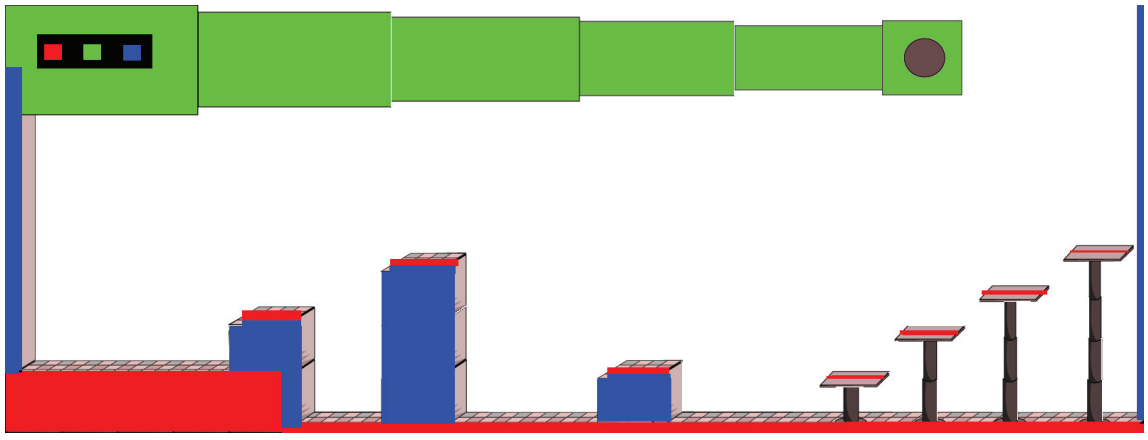
3.1.5.1. Ejemplo

Figura 52. **Ejemplo de un nivel y su mapa de durezas**

Imagen visible



Mapa de durezas



Fuente: propia.

3.1.6. Historia

Dr. Topo es un científico loco que en su laboratorio ha creado una serie de monstruos mutantes y mecánicos como resultado de sus experimentos científicos fallidos. Un día en su laboratorio mientras creaba un robot pollo, este escapa y deshabilita el sistema de control mental de todos sus monstruos que antes obedecían como sirvientes al científico.

Luego de que todas sus creaciones escaparan, su mutante más poderoso lideró una rebelión en su contra para destruirlo. Ahora Dr. Topo debe de recorrer el bosque buscando a sus monstruos para eliminarlos en busca del misterioso *Super* Mutante antes de que estos destruyan todo lo que encuentren a su paso.

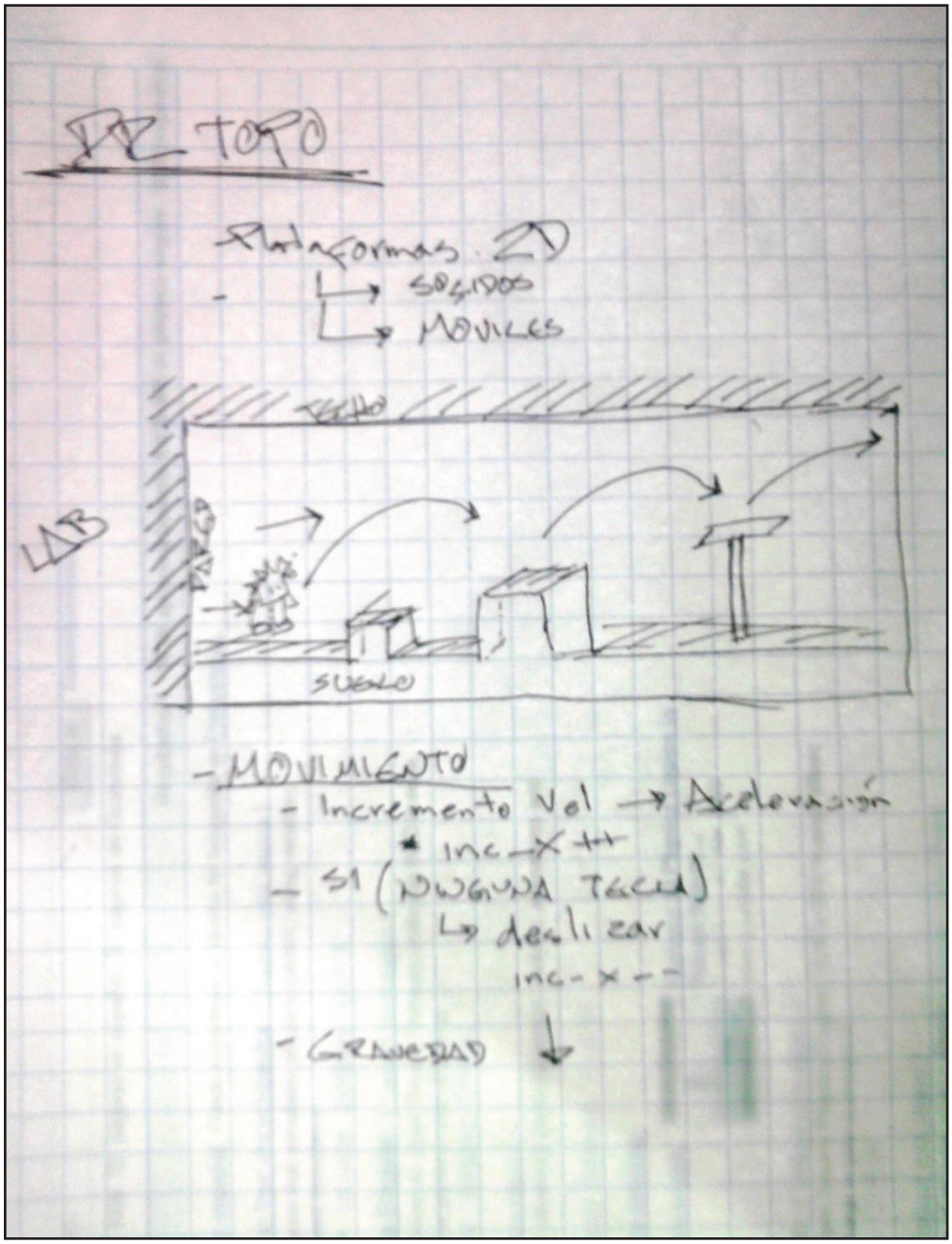
Si Dr. Topo no los encuentra y destruye pronto estos tratarán de eliminarlo por lo que deberás ayudarlo a través de su búsqueda en el bosque hasta llegar al poderoso mutante que antes lo servía y ahora quiere destruirlo.

Figura 53. **Dr. Topo en el primer nivel del videojuego**

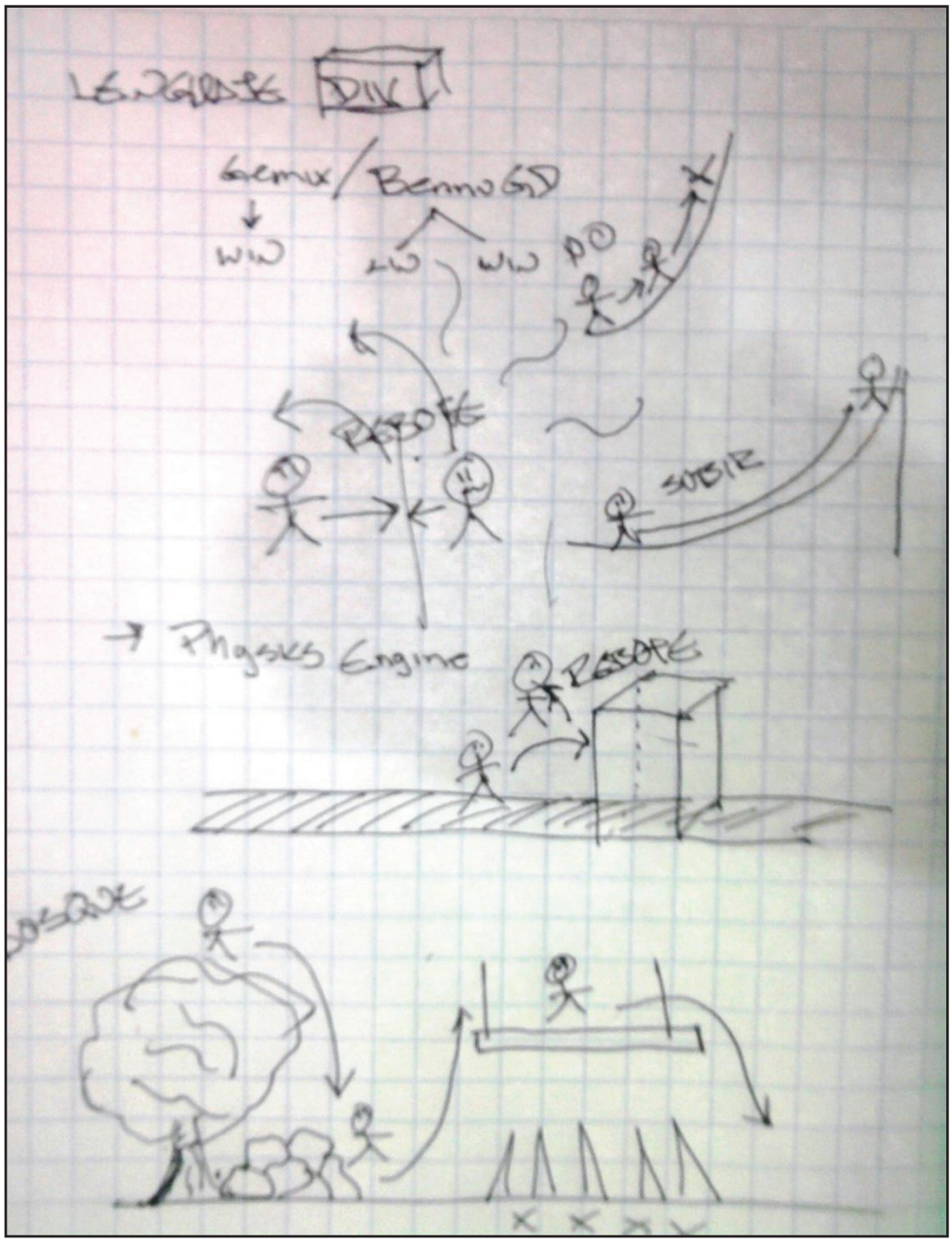
Fuente: propia.

3.1.7. *Story board*

Figura 54. ***Story board* básico del videojuego**

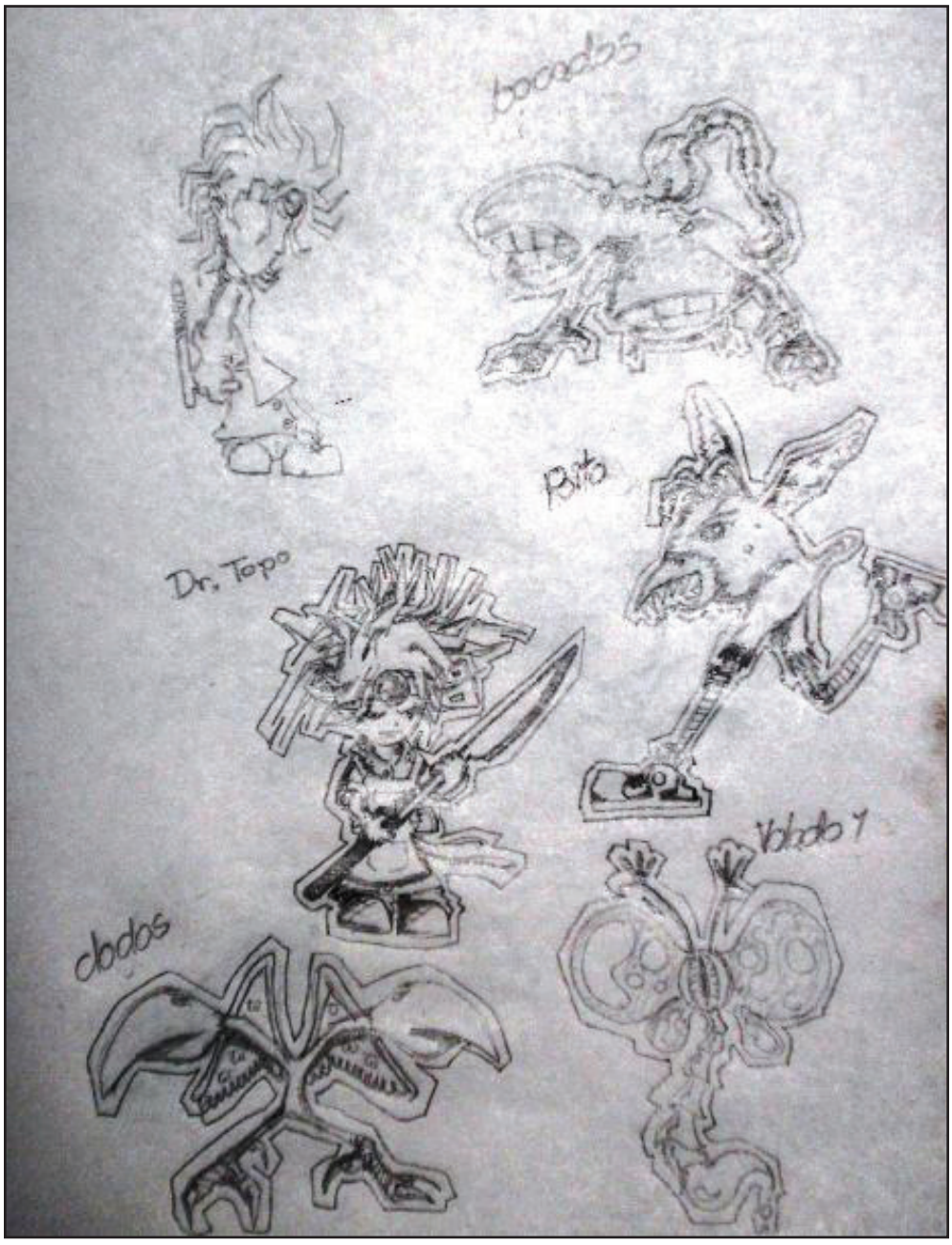


Continúa figura 54



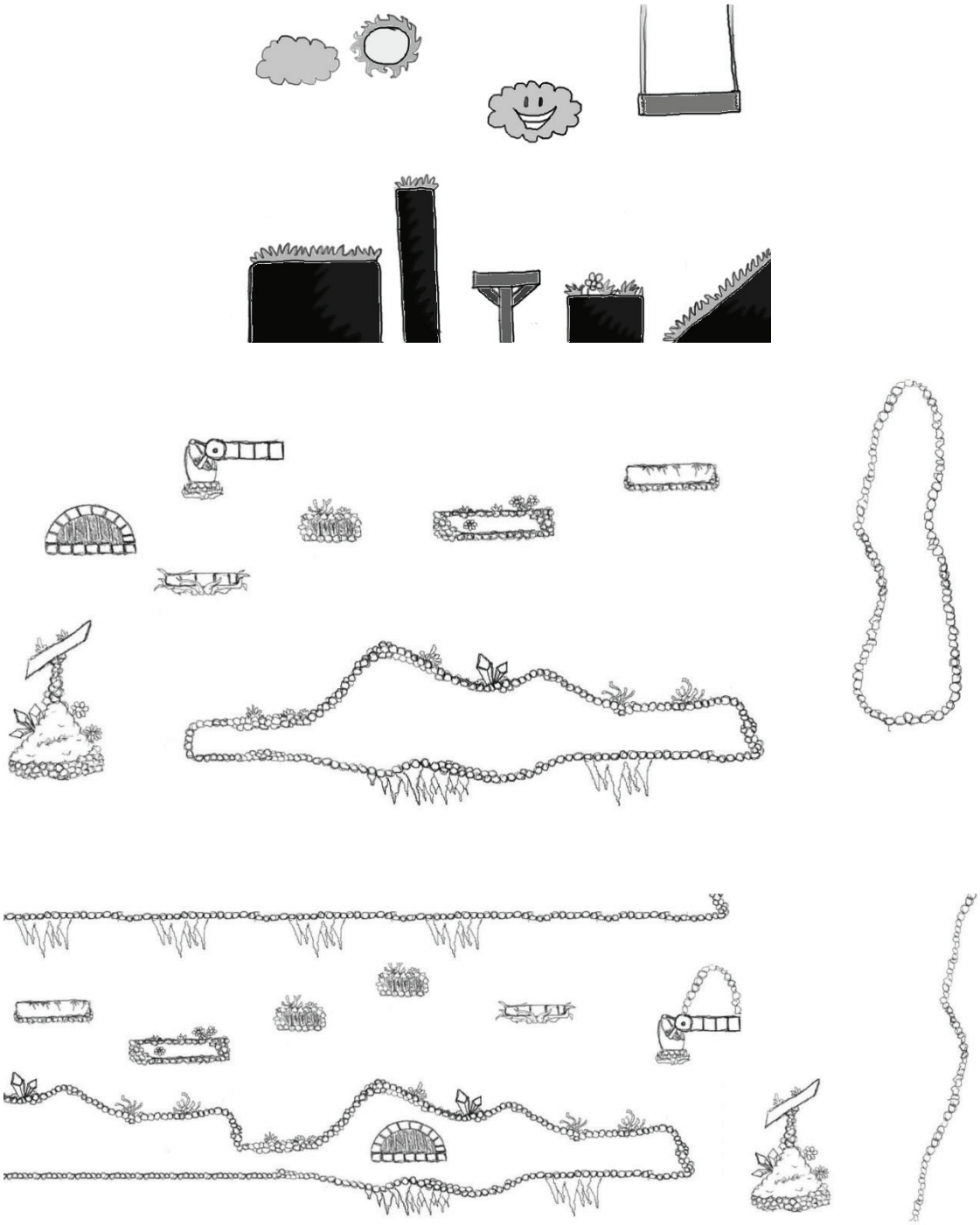
Fuente: propia.

Figura 55. **Ideas para los personajes**



Fuente: propia.

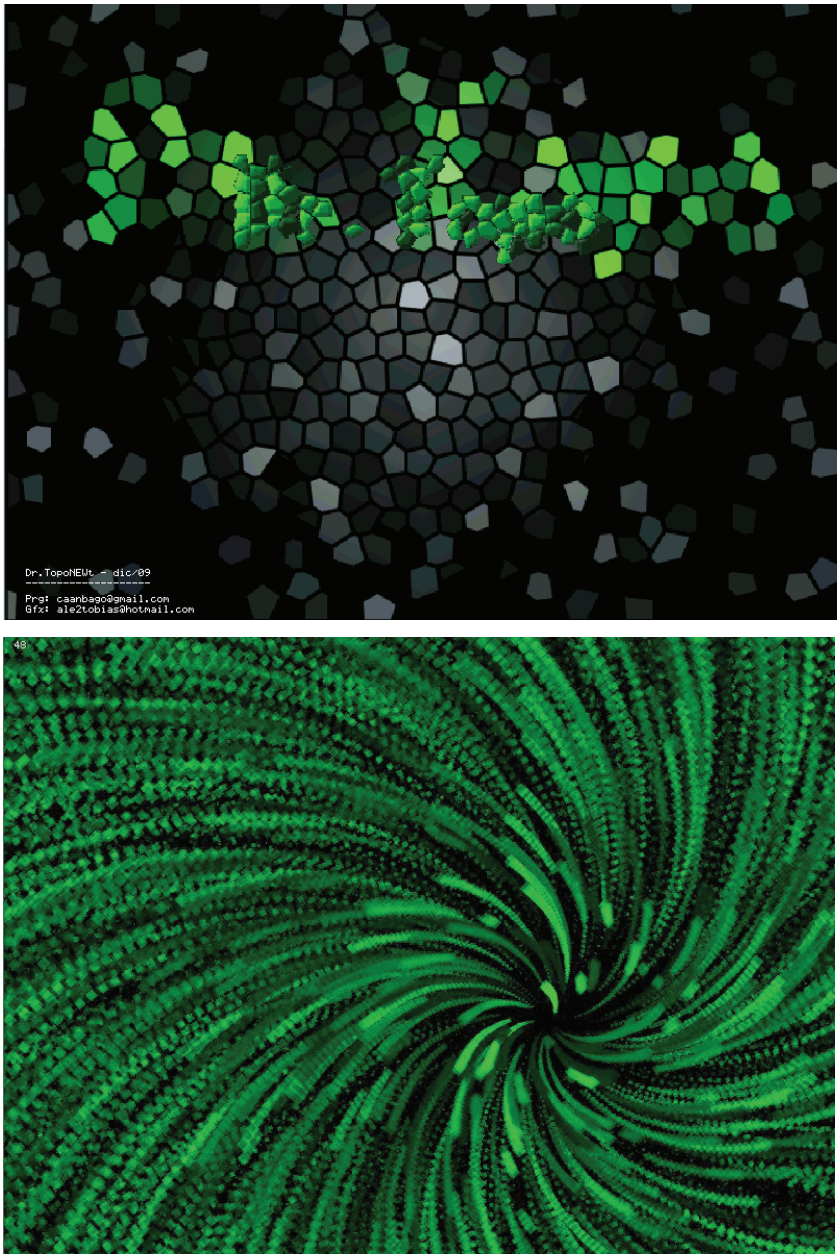
Figura 56. **Ideas para los niveles del videojuego**



Fuente: propia.

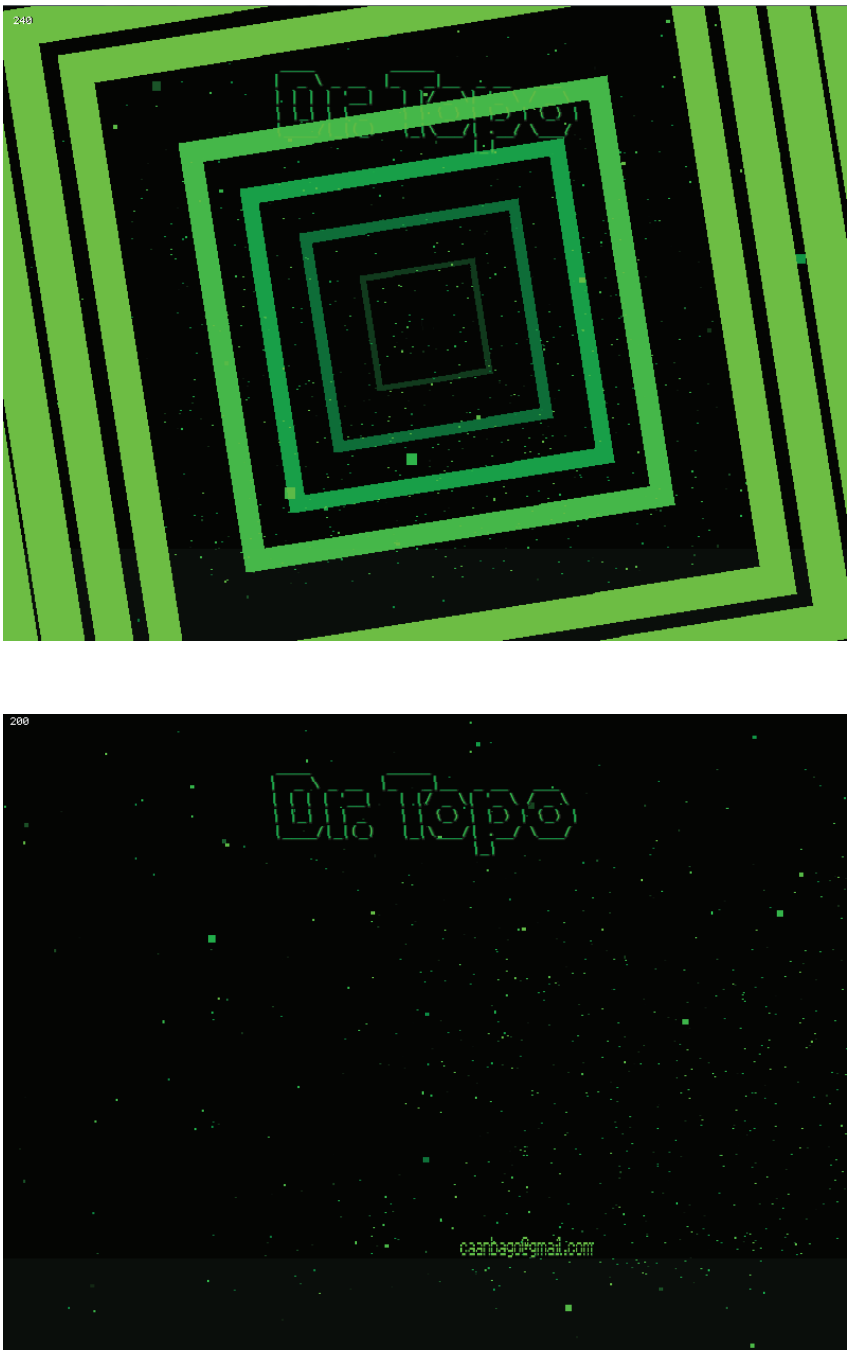
3.1.8. Arte conceptual

Figura 57. **Arte conceptual generada a partir de codificación**



Fuente: propia.

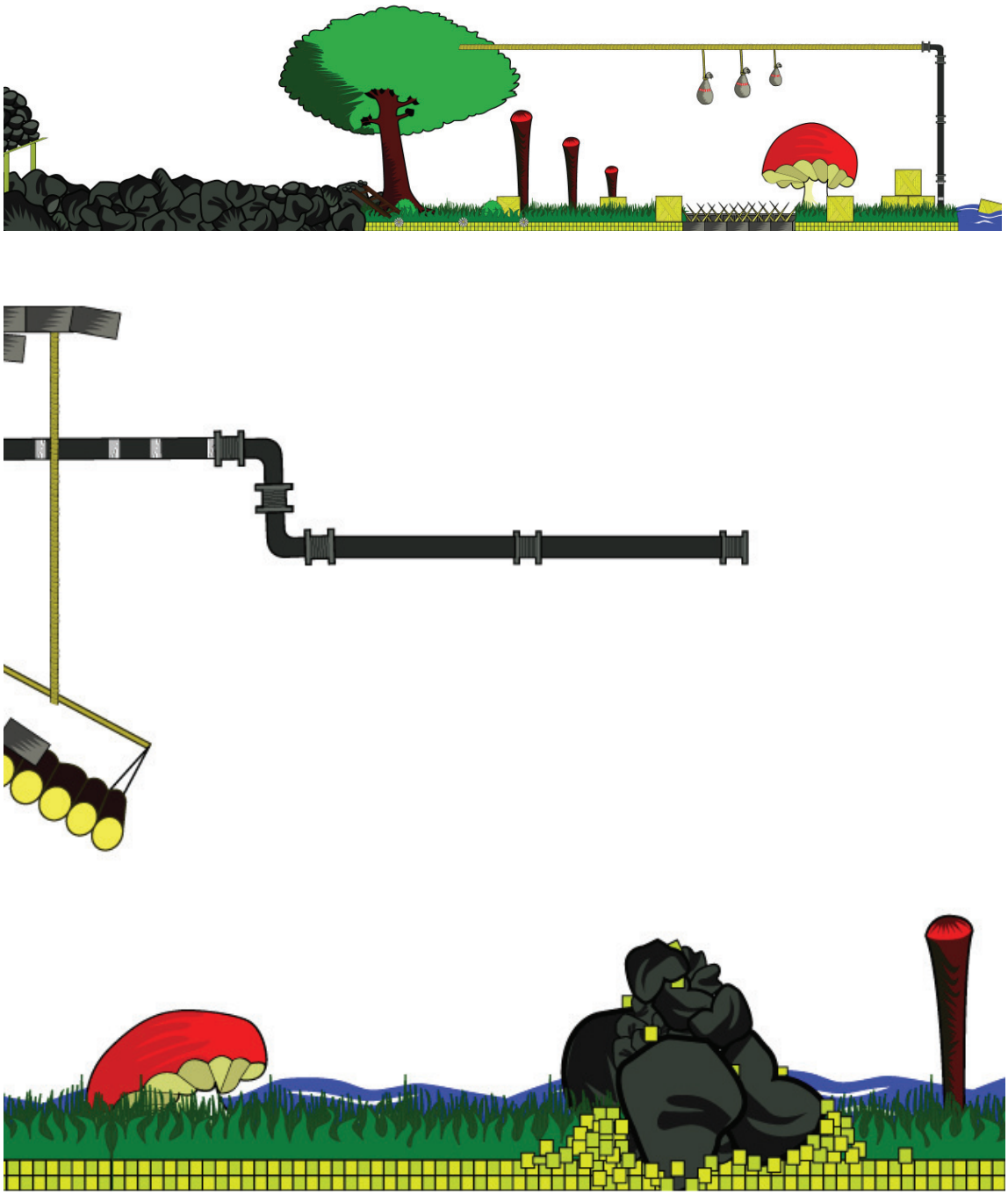
Figura 58. **Arte conceptual inspirada en la *demoscene***



Fuente: propia.

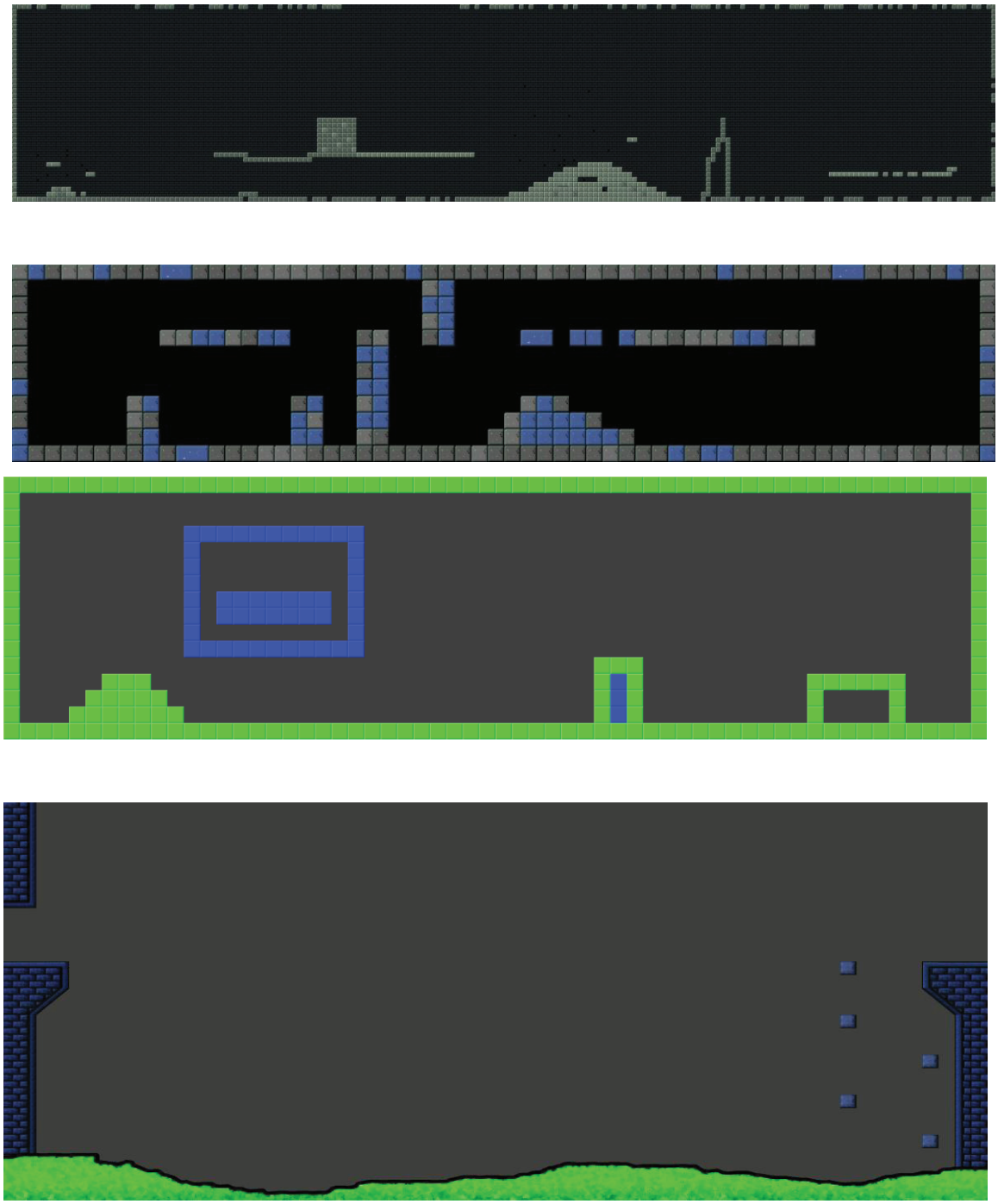
Figura 59. **Arte conceptual de los enemigos del videojuego**

Continúa figura 59



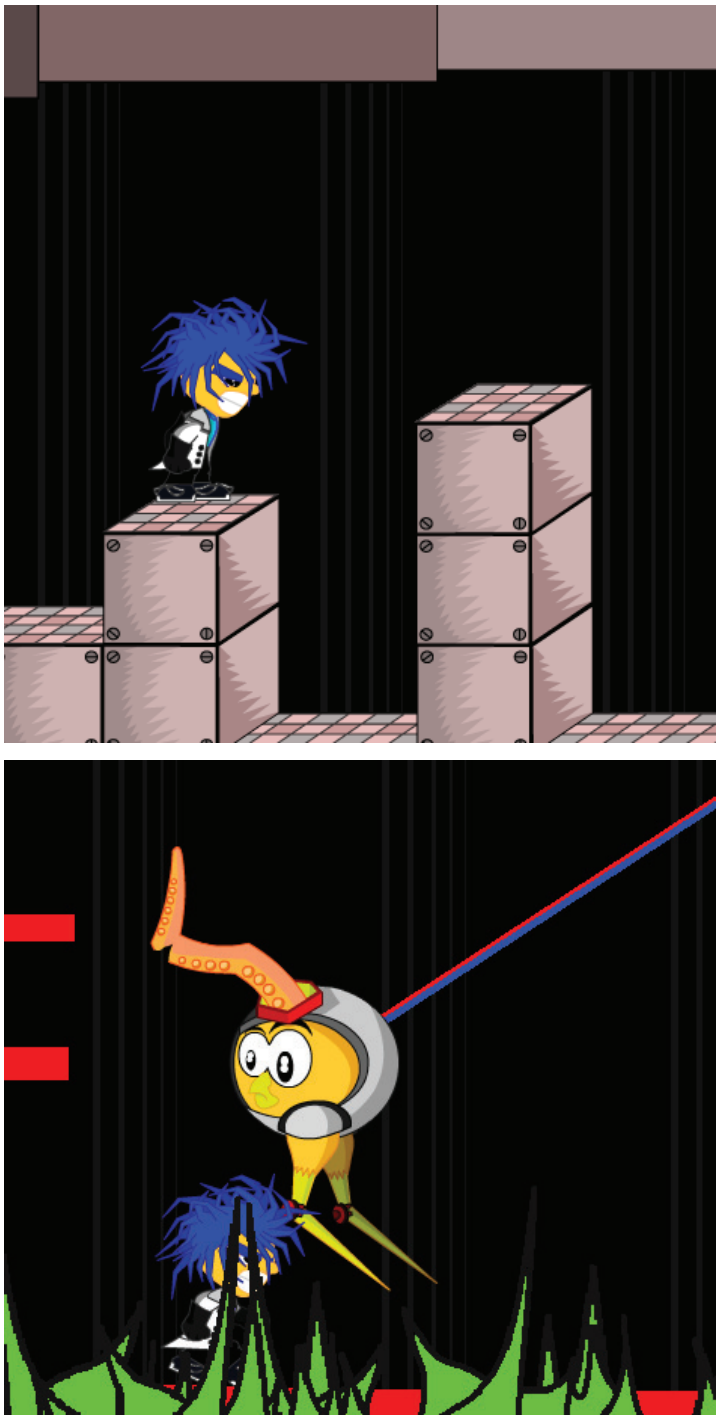
Fuente: propia.

Figura 60. **Arte conceptual de los niveles del videojuego**



Fuente: propia.

Figura 61. **Arte conceptual del videojuego**



Fuente: propia.

3.2. Fase de Producción

3.2.1. Programación del videojuego

A continuación se presenta la explicación de los procesos principales del videojuego. Es importante recordar que el lenguaje de programación Div es un lenguaje orientado a procesos por lo que cada elemento gráfico del videojuego será un proceso independiente que puede ser accedido mediante identificadores de procesos y la jerarquía de procesos del lenguaje.

3.2.2. *Loop* principal del videojuego

En esta sección se inicializan los parámetros para el funcionamiento del programa y se cargan los archivos gráficos, *sprites*, sonidos, etc. necesarios para el funcionamiento global del videojuego.

```
BEGIN

    set_mode(800,600,32);
    set_fps(fps_juego,0);

    //ARCHIVOS FPG
    fpg_topo=load_fpg ("RAT\topo_new.fpg");
    fpg_pollo=load_fpg ("RAT\pollo.rat");
    //fpg_topo=load_fpg ("RAT\topo.rat");

    fpg_intro=load_fpg ("RAT\in.rat");

    //INICIO
    opciones();
    intro();
```

3.2.3. Personaje principal

```
//-----------------------------------------------------------------------------
PROCESS personaje(x,y);
//-----------------------------------------------------------------------------
```

El proceso del personaje es del tipo `c_scroll` para que sus coordenadas `x,y` puedan visualizarse al existir desplazamiento lateral en la pantalla.

```
BEGIN
ctype = C_SCROLL;
size=100;
file=fpg_topo;
graph=50;
```

Animación y desplazamiento a la izquierda:

```
IF(key(_left))
    flags=1; //  izquierda
    graph=ani_camina[ani_t];
    ani_t++;
    IF(ani_t>=sizeof(ani_camina))
      ani_t=0;
    end

    if (incx > -limite_velocidad)
      incx--;
    end
```

Vectores con la secuencia de animación:

```
ani_camina[]     =  1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20;
ani_stand[]      =  50,51,52,53,54,55,56,57,58,59,60,
                    61,63,64,65,66,67,68,69,70,
                    71,72,73,74,75,76,77,78,79,80,
                    81,82,83,84,85,86,87,88,89,90,
                    91,92,93,94,95,96,97,98,99,100,
                    101,102,103,104,105,106,107,108,109;
```

Animación y desplazamiento a la derecha:

```
IF( key(_right))
    flags=0;
    graph=ani_camina[ani_t];
    ani_t++;
    IF(ani_t>=sizeof(ani_camina))
      ani_t=0;
    end //  derecha
    if(incx < limite_velocidad)
      incx++;
    end
```

Al no presionar ninguna tecla, el personaje pasa a estar sin acción:

```
IF((!key(_left) AND !key(_right)) OR (key(_left) AND  key(_right)))
    graph=ani_stand[ani_t];
    ani_t++;
    IF(ani_t>=sizeof(ani_stand))
      ani_t=0;
    end
        IF (incx < 0)
                incx+=4/3;//incx++;
           END
           IF (incx > 0)//incx--;
          incx-=4/3;
           END
           if(incx ==0 or incx == 1 or incx ==-1)
              incx=0;
           end//   INCX=0;
  END
```

Salto del personaje:

```
IF((key(_up) or key(_a))  AND en_suelo)
  velocidad_gravedad=-fuerza_salto-(abs(incx/5));
end

if(vida<=0) Size--; end;
```

3.2.4. Proceso para detección de paredes

Este proceso se utiliza para detectar las paredes del mapa de durezas. Los colores de durezas son:

Tabla VI. **Códigos de dureza para las paredes**

COLOR	RGB
Azul	0,0,255
Blanco	255,255,255

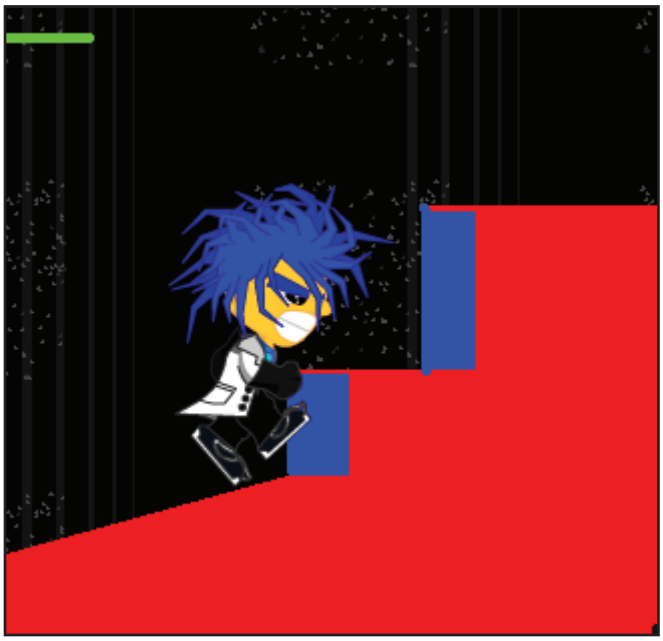
Fuente: propia.

```
color_pared=rgb(0,0,255);
color_pared_blanco=rgb(255,255,255);


IF (father.incx <> 0)

    contx=father.incx/abs(father.incx);

        FOR (new_incx=0; new_incx <> father.incx ; new_incx=new_incx + contx)
            IF (map_get_pixel (id_fpg_fondo,id_map_fondo, father.x + new_incx + contx, father.y + alto /2 ) == color_pared
             OR map_get_pixel (id_fpg_fondo,id_map_fondo, father.x + new_incx + contx, father.y +  alto /4) == color_pared
             OR map_get_pixel (id_fpg_fondo,id_map_fondo, father.x + new_incx + contx, father.y  ) == color_pared
             OR map_get_pixel (id_fpg_fondo,id_map_fondo, father.x + new_incx + contx, father.y - alto /4) == color_pared
             OR map_get_pixel (id_fpg_fondo,id_map_fondo, father.x + new_incx + contx, father.y - alto /2) == color_pared
             //blanco
             or map_get_pixel (id_fpg_fondo,id_map_fondo, father.x + 20 + new_incx + contx, father.y + alto /2 ) == color_pared_blanco
             OR map_get_pixel (id_fpg_fondo,id_map_fondo, father.x + 20 +  new_incx + contx, father.y +  alto /4) == color_pared_blanco
             OR map_get_pixel (id_fpg_fondo,id_map_fondo, father.x + 20 +  new_incx + contx, father.y  ) == color_pared_blanco
             OR map_get_pixel (id_fpg_fondo,id_map_fondo, father.x - 20 +  new_incx + contx, father.y ) == color_pared_blanco
             OR map_get_pixel (id_fpg_fondo,id_map_fondo, father.x + 20 +  new_incx + contx, father.y - alto /2) == color_pared
             )

              if(map_get_pixel (id_fpg_fondo,id_map_fondo, father.x + new_incx + contx, father.y  ) <> rgb(255,0,0))
                father.incx=-(father.incx/2);
              end
                BREAK;
            else
          //father.SHEAR_X= 0;
          //father.SHEAR_y= 0;
END
```

La función `map_get_pixel(<fichero>, <gráfico>, <x>, <y>)` obtiene el color del pixel en el mapa de durezas dependiendo de la posición del proceso padre. Para ello requiere el `<código del fichero>` en el que se encuentra el gráfico, el `<código del gráfico>` dentro del fichero y las coordenadas `(x, y)` del punto del gráfico cuyo color se quiere obtener.

```
if(map_get_pixel (id_fpg_fondo,id_map_fondo, father.x + new_incx + contx, father.y  ) <> rgb(255,0,0))
  father.incx=-(father.incx/2);
end
```

Figura 62. **Durezas en el nivel de prueba**



Fuente: propia.

3.2.5. Proceso para detección del techo y el suelo

Este proceso se utiliza para detectar el techo y el suelo del mapa de durezas. Los colores de durezas son:

Tabla VII. **Códigos de dureza para el suelo y el techo**

COLOR SUELO	RGB
Rojo	255,0,0
Blanco	255,255,255

Continúa Tabla VII

COLOR TECHO	RGB
Verde	0,0,255
Blanco	255,255,255

Fuente: propia.

Detección del techo:

```
IF (gravedad_temp<0)

    WHILE (gravedad_temp++!=0)
        IF (map_get_pixel(id_fpg_fondo,id_map_fondo,father.x,(father.y-offset_sup))<>color_techo
           and map_get_pixel(id_fpg_fondo,id_map_fondo,father.x,(father.y-offset_sup))<>color_pared_blanco
         )
           father.y--;
        ELSE
            father.velocidad_gravedad=0;
            BREAK;
        END
    END
```

Detección del suelo:

```
  //suelo
father.y+=gravedad_temp;

   FROM gravedad_temp=-15 TO 8;
      IF (map_get_pixel(id_fpg_fondo,id_map_fondo,father.x,(father.y+gravedad_temp+offset_inf))==color_suelo
        or map_get_pixel(id_fpg_fondo,id_map_fondo,father.x+15,(father.y+gravedad_temp+offset_inf))==color_suelo
        or map_get_pixel(id_fpg_fondo,id_map_fondo,father.x-15,(father.y+gravedad_temp+offset_inf))==color_suelo

        or map_get_pixel(id_fpg_fondo,id_map_fondo,father.x,(father.y+gravedad_temp+offset_inf))==color_pared_blanco
        or map_get_pixel(id_fpg_fondo,id_map_fondo,father.x+15,(father.y+gravedad_temp+offset_inf))==color_pared_blanco
        or map_get_pixel(id_fpg_fondo,id_map_fondo,father.x-15,(father.y+gravedad_temp+offset_inf))==color_pared_blanco
       )
          father.en_suelo=TRUE;  BREAK;
      END
   END
```

Figura 63. **Durezas de suelo y techo**

Fuente: propia.

3.2.6. *Map tiler* y edición de *tile maps*

3.2.6.1. Algoritmo

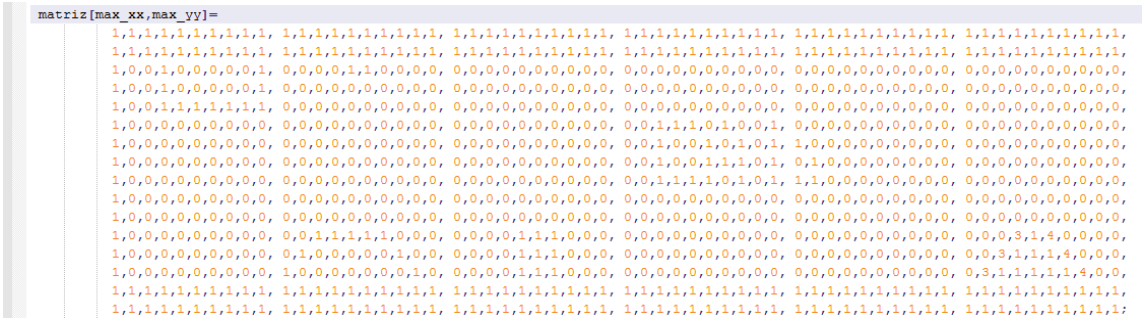
Este algoritmo permite crear los mapas de algunas pantallas mediante una matriz que representa el mapa que se generará dinámicamente.

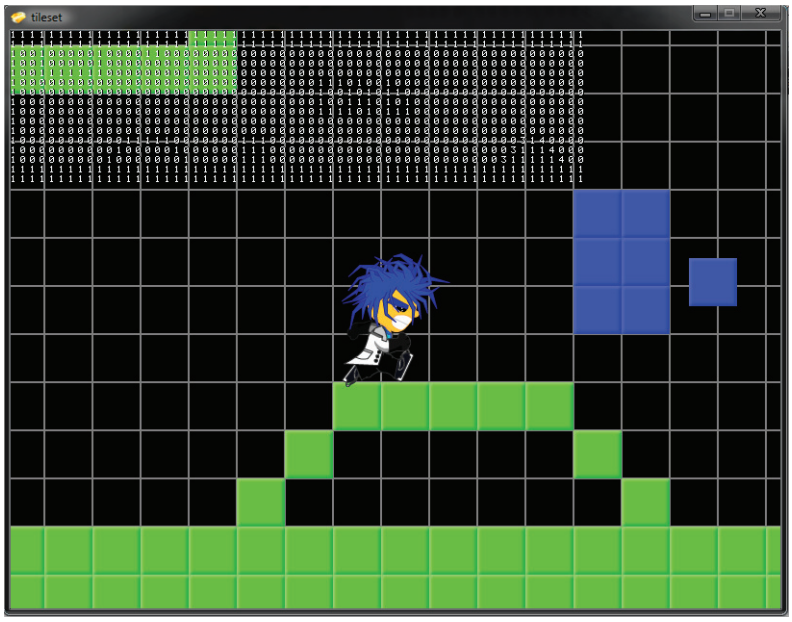
- ✓ `matriz[ancho,alto]`

3.2.6.2. Ejemplo

La matriz contiene el código de *tile* que se colocará en el nuevo mapa generado. El código 0 representa un *tile* transparente.

Figura 64. **Matriz y nivel generado por el algoritmo del *map tiler***





Fuente: propia.

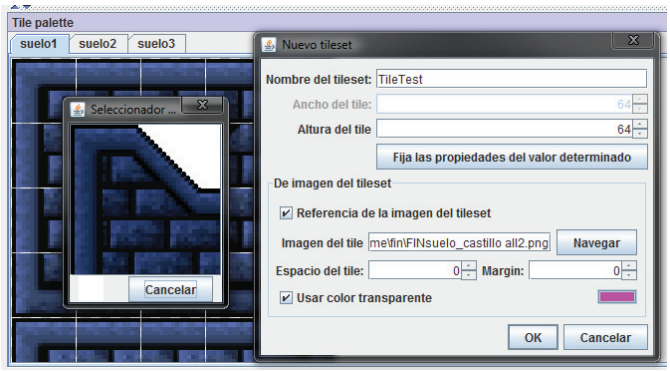
El algoritmo básico para *tiles* de 50 pixeles es:

```
for(largo=0; largo<=max_xx; largo++)
  for(alto=0; alto<=max_yy; alto++)
    write_int(0,largo*10,alto*10,0,&matriz[largo,alto]);

    tilex=largo*50;
    tiley=alto*50;
    if(matriz[largo,alto]<>0)
      tile_a=matriz[largo,alto];
    else
      tile_a=999;
    end

    map_put(fpg_tile, pantalla, tile_a, tilex, tiley);
  end
end
```

3.2.7. Edición de mapas utilizando *Tiled Map Editor*

Mediante esta herramienta se editaron algunos de los niveles del videojuego. Los *tiles* fueron creados utilizando la técnica de *pixel art* en mapas de 64x64 pixeles de tamaño y luego fueron importados al programa.

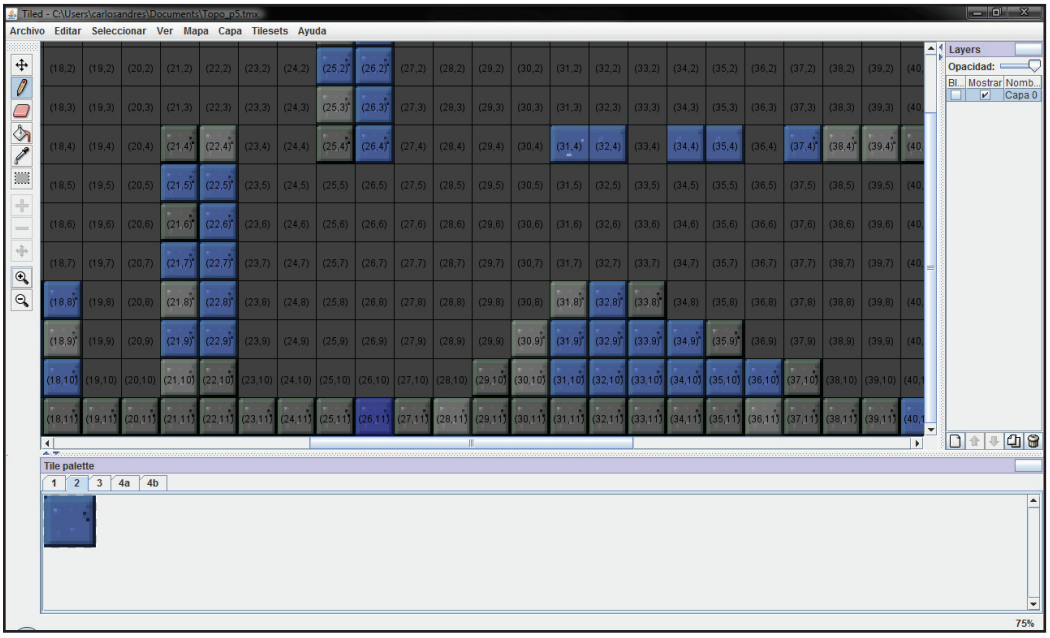
Figura 65. **Carga de *tiles* en *Tiled Map Editor***



Fuente: propia.

Luego de importar los *tiles* al programa se edita el mapa colocando cada *tile* en la coordenada especificada y por último es exportado el nivel que se utilizará en el videojuego.

Figura 66. **Edición de niveles utilizando *Tiled Map Editor***



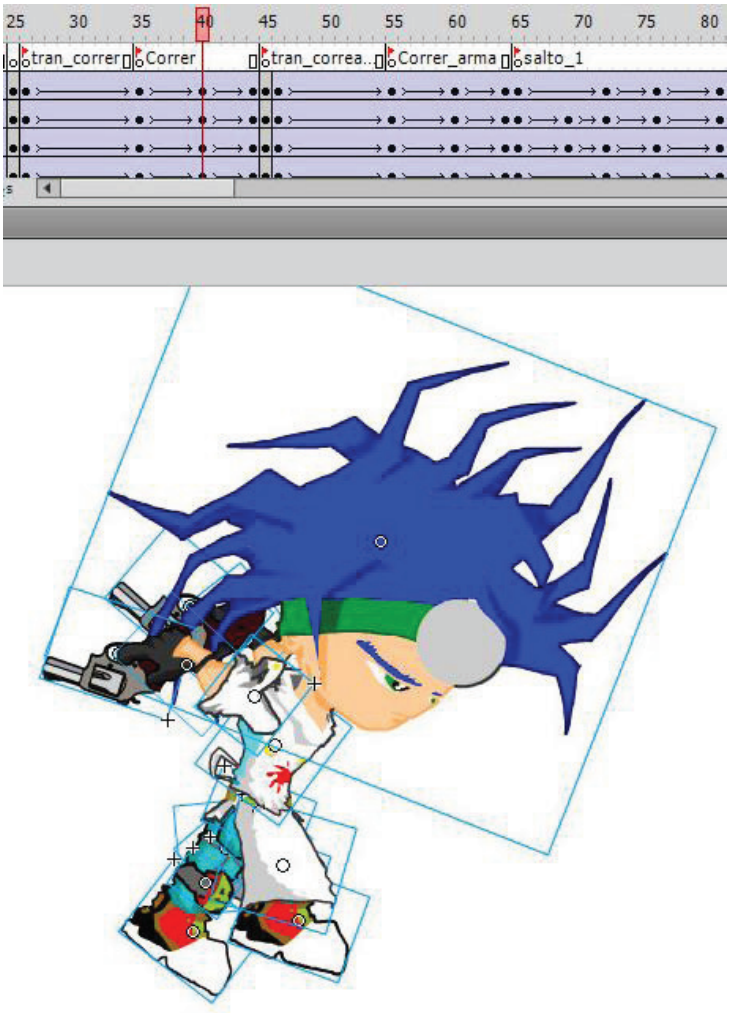
Fuente: propia.

3.2.8. Diseño gráfico

Para la fase de diseño gráfico se crearon gráficos vectoriales para facilitar el proceso de animación, manipulación y copiado de imágenes utilizando capas con canal *alpha* para las secciones transparentes y exportando *sprites* de 32 *bits*.

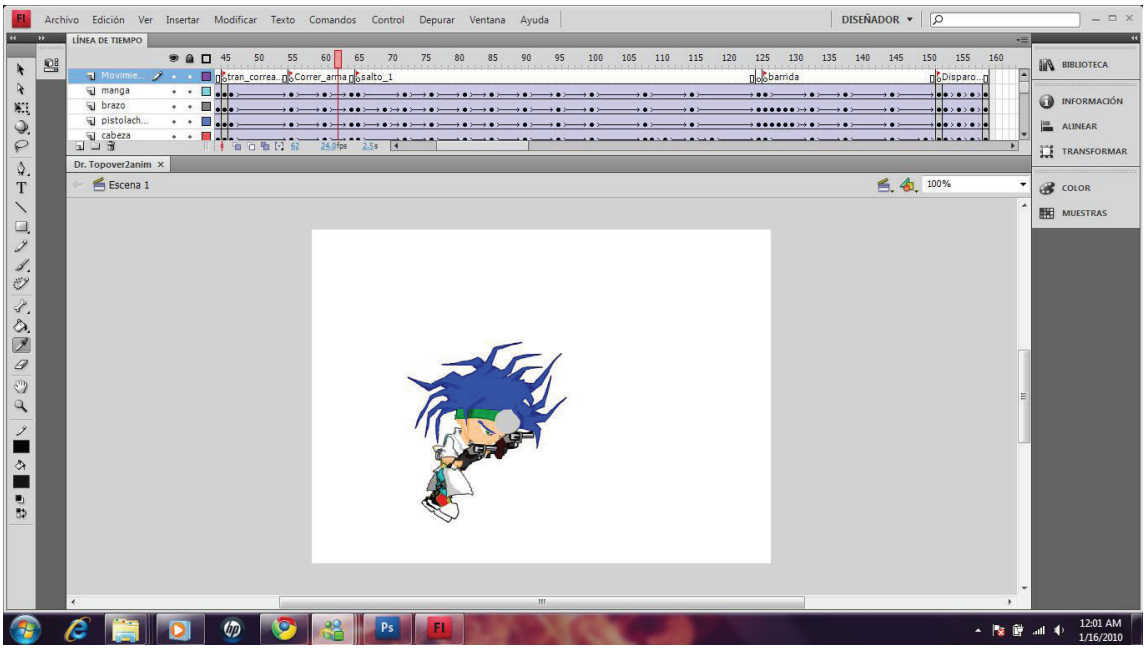
3.2.8.1. Animaciones

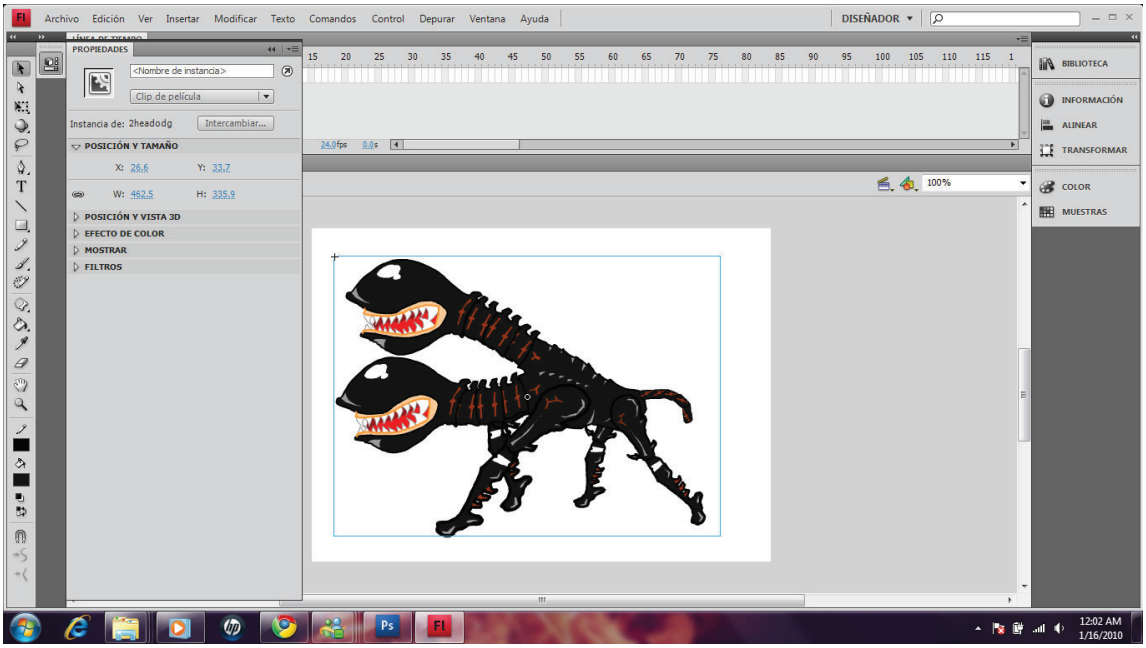
Figura 67. **Animación del personaje principal**



Fuente: propia.

Figura 68. **Edición de las animaciones**

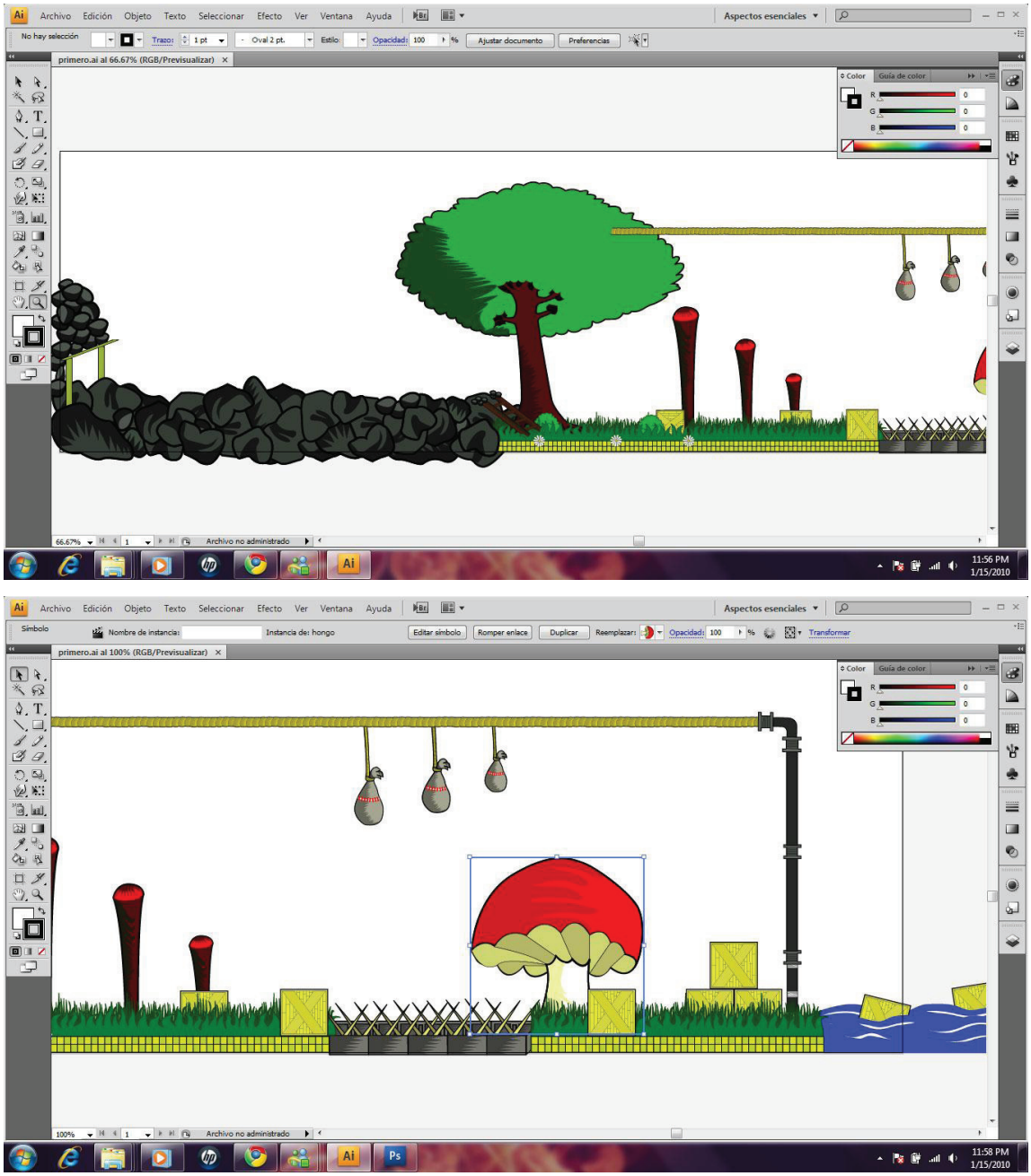




Fuente: propia.

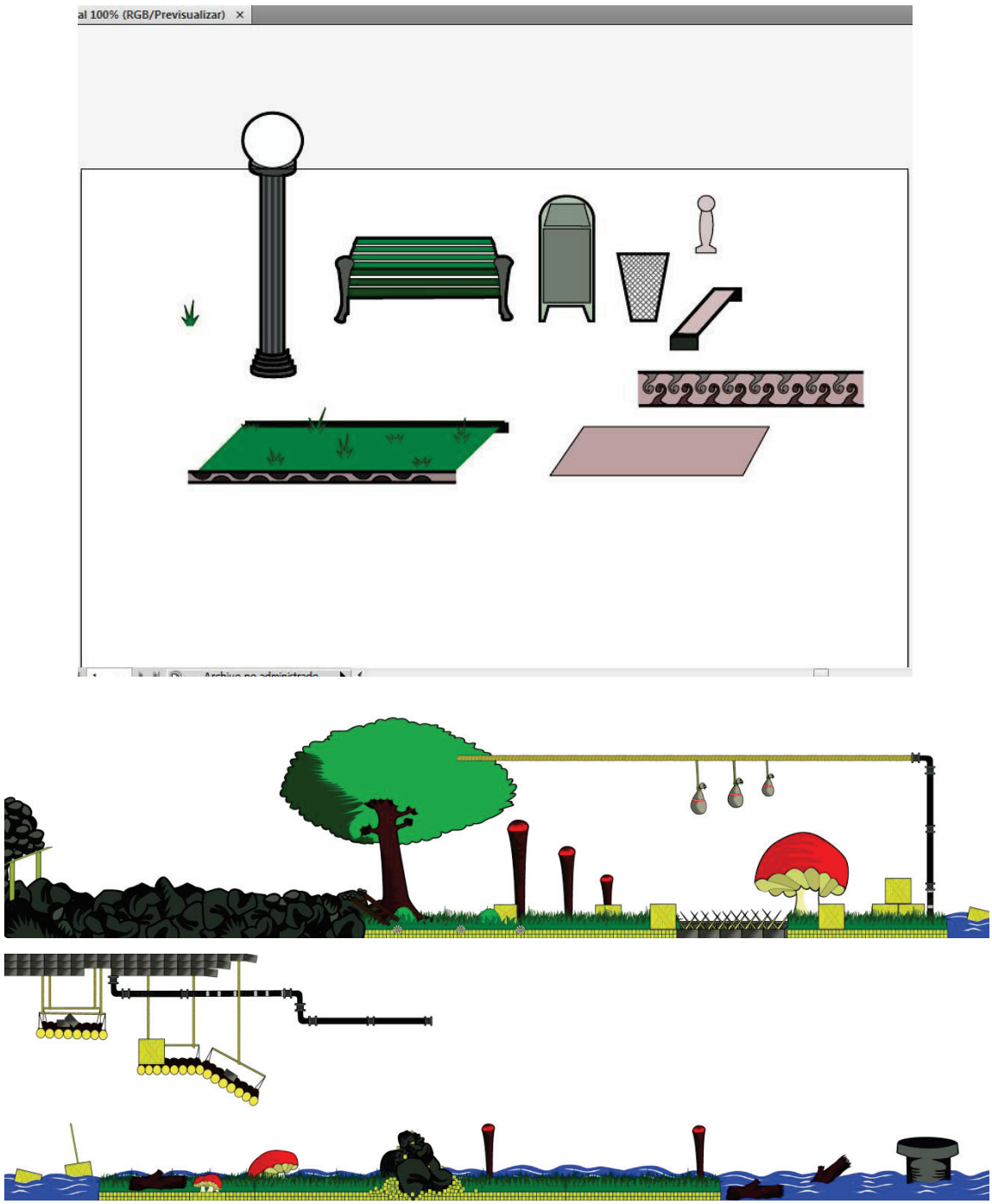
3.2.8.2. Niveles y otros elementos

Figura 69. **Edición de niveles**



Fuente: propia.

Figura 70. **Edición elementos para los niveles**

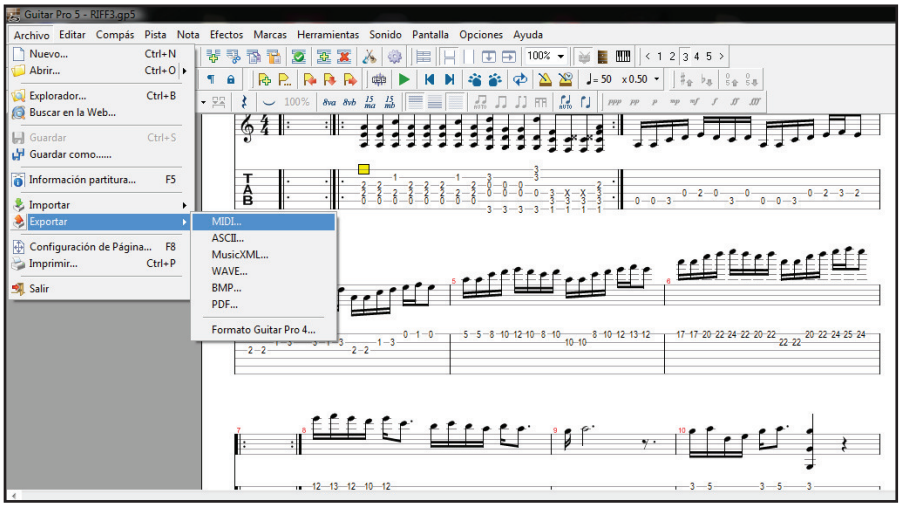


Fuente: propia.

3.2.9. Edición de efectos de sonido y música

El proceso de creación de la música y sonidos para el videojuego se inicia con la composición de el tema musical, luego de esto se escribe la partitura o tablatura utilizando los programas *Guitar Pro* o *TuxGuitar*. Al terminar la partitura, esta se exporta a formato MIDI para poder ser editada utilizando FL *Studio* o *MadTracker* que por último se exportará a formato ogg o mp3.

Figura 71. **Edición de partitura**

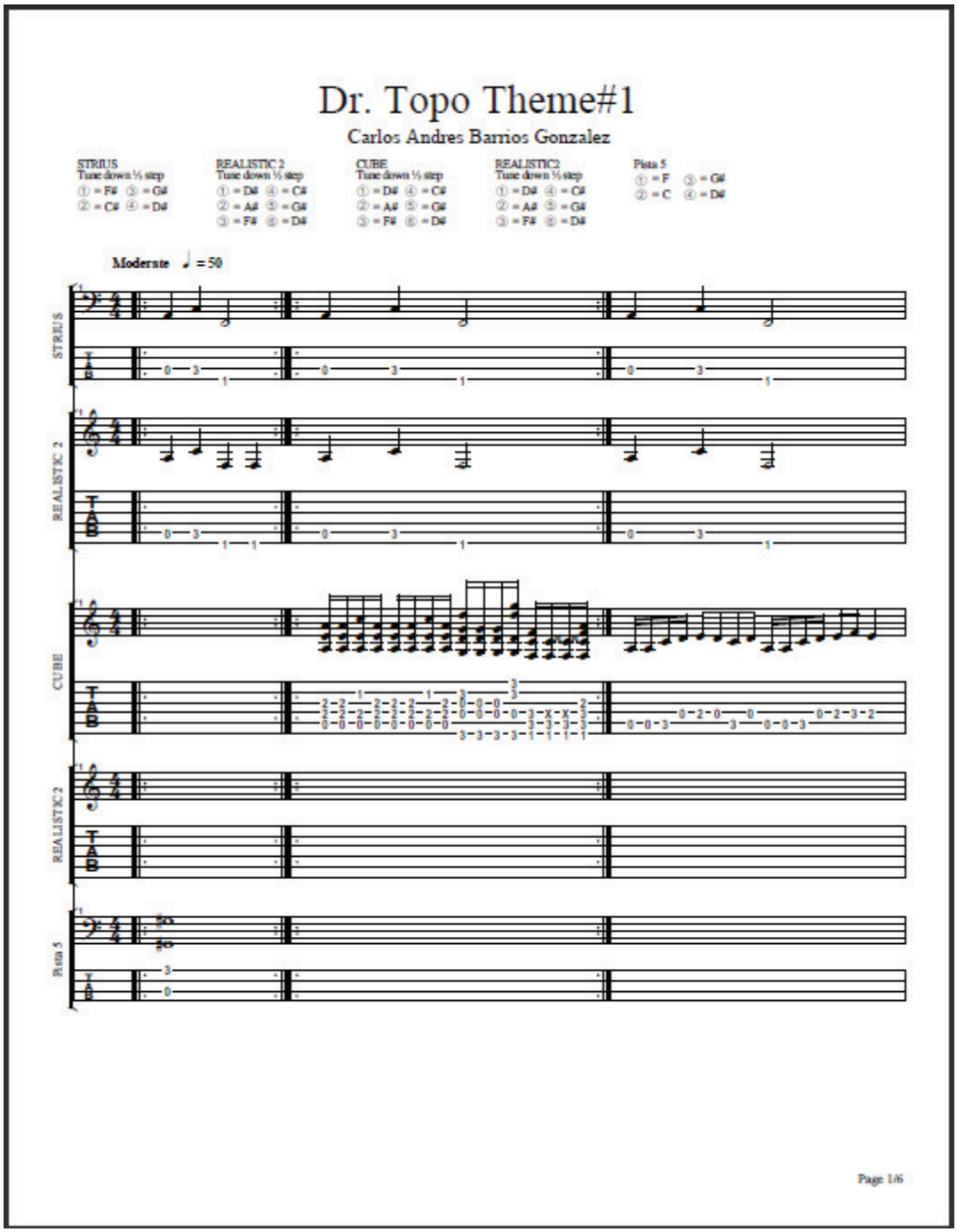


Fuente: propia.

3.2.9.1. Escritura de la partitura

- ✓ La partitura debe de cumplir las reglas de estructura (pentagramas, notas, figuras, alteraciones, armaduras de clave, claves, compas y otros componentes);
- ✓ Cada instrumento es separado por pista;
- ✓ Al finalizar la partitura esta se exporta a formato MIDI (*Musical Instrument Digital Interface*);

Figura 72. **Partitura de un tema del videojuego**



Fuente: propia.

✓ El archivo MIDI es importado a la estación de audio digital FL *Studio*;

Figura 73. **Edición de pistas en FL *Studio***

Fuente: propia.

✓ Cada canal importado puede convertirse en un *Sampler* o VSTi;
✓ Posteriormente se aplican los efectos con los *plugins* disponibles;
✓ Los sintetizadores utilizados son en el proyecto son:
  - ymVST
  - Sytrus

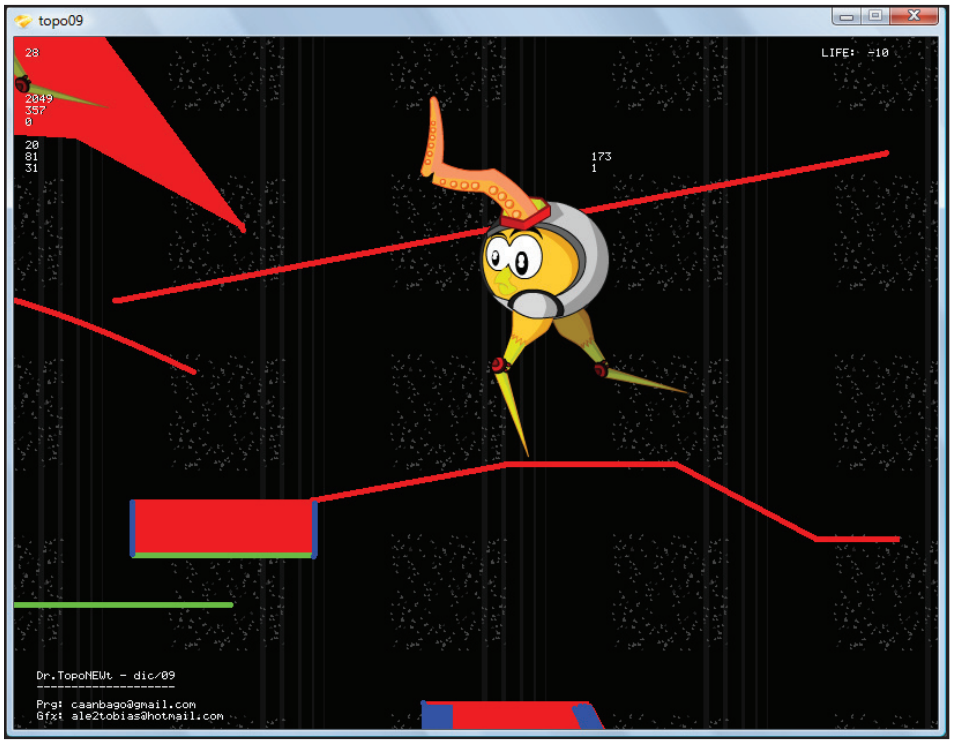
✓ Por último se renderiza la pista y es exportada a formato ogg o mp3.

3.2.10. Fase de pruebas

Las pruebas *alpha* se inician desde que el programa es jugable y son realizadas por las personas involucradas en el desarrollo del videojuego para corregir cualquier defecto grave y mejorar las características de jugabilidad.

Las pruebas realizadas a las versiones *alpha* son notificadas mediante correo electrónico o chat para realizar las modificaciones y mejoras necesarias en el programa.

Figura 74. **Versión *alpha* de Dr. Topo**



Fuente: propia.

Luego de la publicación de la primera versión beta en la página web del videojuego, toda notificación de error o sugerencias se recibirá por medio de correo electrónico.

3.2.10.1. Pruebas *alpha*

Las pruebas *alpha* han sido realizadas por el equipo de desarrollo luego de alcanzar un hito o cambio mayor. El equipo de pruebas *alpha* está formado por:

- ✓ Carlos Andrés Barrios González
- ✓ José Alejandro Tobias Cruz

3.2.10.2. Pruebas beta

Las pruebas beta han sido realizadas en mayor parte por usuarios del foro de Gemix *Studio* luego de la publicación de la primera beta pública. El enlace del foro es: http://gemixstudio.com/forums/viewtopic.php?f=72&t=1155

3.2.10.3. *Betatesters* oficiales

- ✓ CicTec (principal desarrollador del compilador Gemix y líder del proyecto)
- ✓ Ing. Rafe Kaplan (Ingeniero de *Software* en Google Inc. y miembro del grupo de desarrollo del API Google App *Engine*)
- ✓ Ing. Julio Lam
- ✓ Nower
- ✓ GINO
- ✓ Erkosone
- ✓ Nightwolf
- ✓ Dluk
- ✓ TYCO
- ✓ Martindamiano

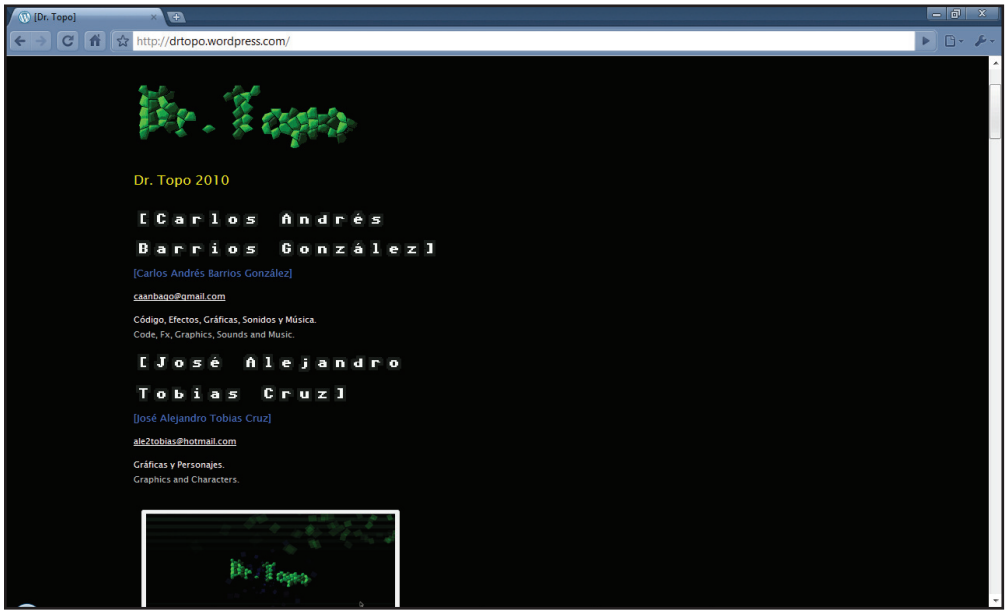
3.2.11. Publicación

El videojuego se publicó mediante una página web que contiene los enlaces de descarga de la última versión disponible. Al ser un videojuego independiente gratuito no es necesario contar con ninguna empresa dedicada a la distribución de *software* y este puede ser descargado por cualquier persona.

3.2.11.1. Página del videojuego

✓ Enlace: http://drtopo.freeiz.com/

Figura 75. **Página principal del videojuego**

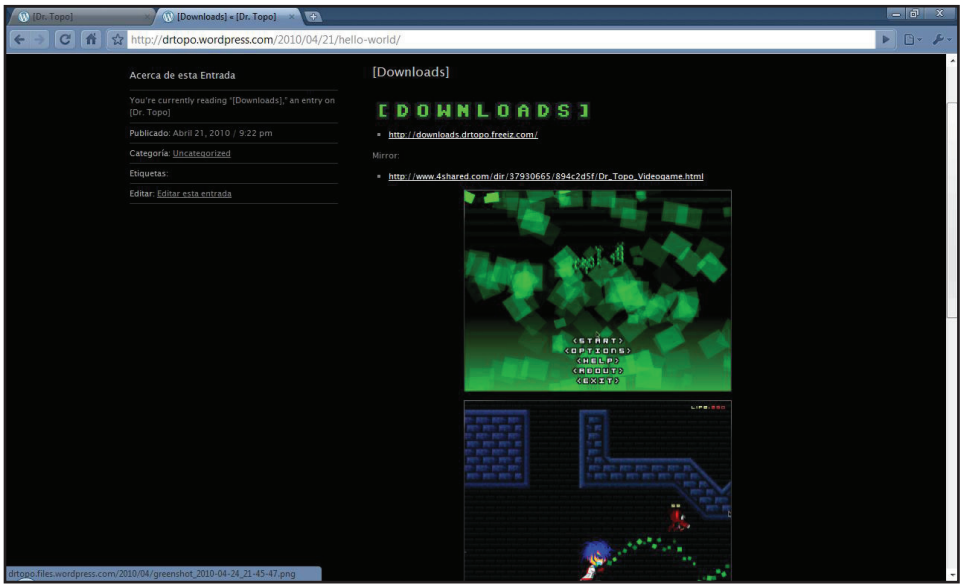


Fuente: propia.

3.2.11.2. Enlaces de descarga

- ✓ http://www.4shared.com/dir/37930665/894c2d5f/Dr_Topo_Videogame.html
- ✓ http://downloads.drtopo.freeiz.com/

Figura 76. **Página de descargas**



Fuente: propia.

3.2.11.3. Información de contacto

- ✓ Carlos Andrés Barrios González
- ✓ Correo electrónico: caanbago@gmail.com

3.3. Resultados finales

3.3.1. Evaluación de los usuarios

La evaluación de Dr. Topo se ha realizado tomando como base los criterios que generalmente son evaluados en las revistas dedicadas a los videojuegos. De 21 usuarios que completaron una encuesta se han obtenido los siguientes resultados, siendo 10 el valor más alto de la escala y 0 el más bajo.

Tabla VIII. **Datos recolectados de los usuarios**

	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T14	T14	T15	T16	T17	T18	T19	T20	T21	PROMEDIO
Gráficos																						
Calidad gráfica	9	10	9	10	8	9	9	7	9	9	10	9	9	9	7	9	8	9	10	9	9	8.904761905
Animaciones	8	9	9	10	9	9	7	8	10	8	9	10	9	10	8	9	9	9	10	8	9	8.904761905
Detalles	10	10	10	9	9	10	9	9	10	9	10	9	9	9	8	9	9	9	9	9	9	9.238095238
Mapas	8	10	9	8	9	8	9	8	7	9	7	8	9	10	9	8	9	8	9	9	8	8.523809524
Efectos	10	9	9	10	10	10	9	9	8	10	9	10	10	10	9	9	9	10	9	10	10	9.476190476
Jugabilidad																						
Gameplay	8	7	9	9	8	9	8	9	9	8	9	8	7	8	7	9	6	8	9	7	8	8.095238095
Temática	10	9	9	10	9	10	9	9	8	7	9	10	8	9	9	9	8	10	10	10	10	9.142857143
Uso de teclado/joystic	10	9	9	10	9	10	10	10	10	8	10	9	10	9	8	9	10	10	10	9	10	9.476190476
Sonido y Música:	10	10	9	10	9	10	10	8	10	10	9	10	8	9	10	9	10	10	9	9	10	9.476190476
PROMEDIO	9.2	9.2	9.1	9.6	8.9	9.4	8.9	8.6	9	8.7	9.1	9.2	8.8	9.2	8.3	8.9	8.7	9.2	9.4	8.9	9.2	**9.026455026**

Fuente: propia.

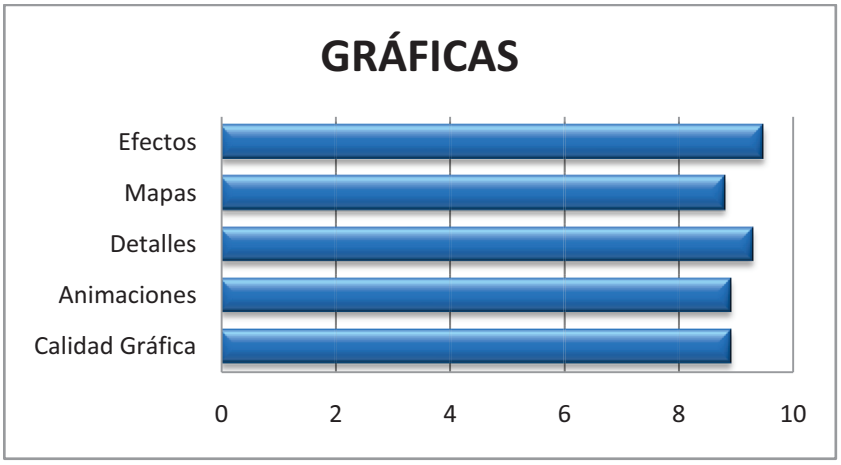
3.3.2. Tablas y gráficos

Tabla IX. **Promedio de resultados de la sección "Gráficos"**

GRÁFICOS	
Calidad gráfica	8.9047619
Animaciones	8.9047619
Detalles	9.28571429
Mapas	8.80952381
Efectos	9.47619048
Resultado	9.07619048

Fuente: propia.

Figura 77. **Gráfica con los resultados de la sección "Gráficos"**



Fuente: propia.

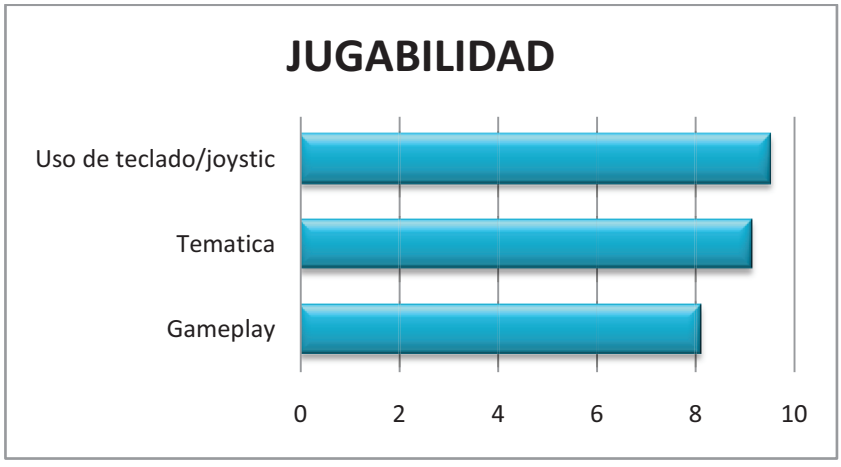
3.3.3. Jugabilidad

Tabla X. **Promedio de resultados de la sección "Jugabilidad"**

JUGABILIDAD	
Game play	8.0952381
Temática	9.14285714
Uso de teclado/*joystic*	9.52380952
Resultado	8.92063492

Fuente: propia.

Figura 78. **Gráfica con los resultados de la sección "Jugabilidad"**



Fuente: propia.

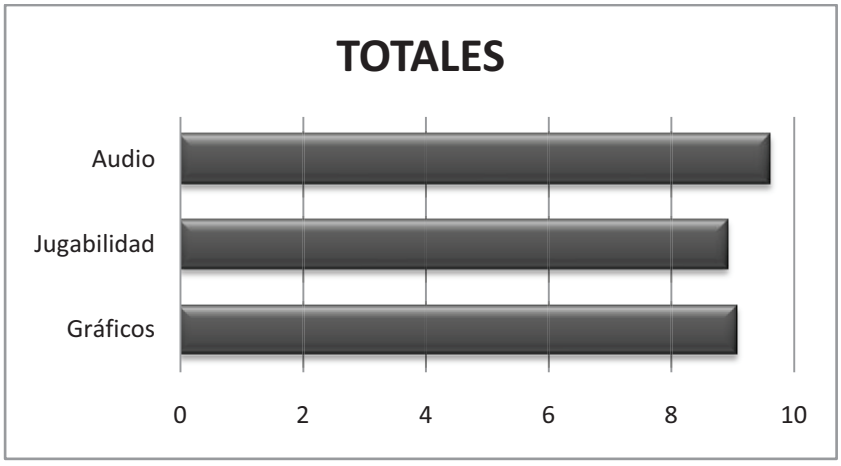
3.3.4. Promedio de resultados

Tabla XI. **Promedio de resultados totales**

Gráficos	9.07619048
Jugabilidad	8.92063492
Audio	9.61904762
TOTAL	**9.02645503**

Fuente: propia.

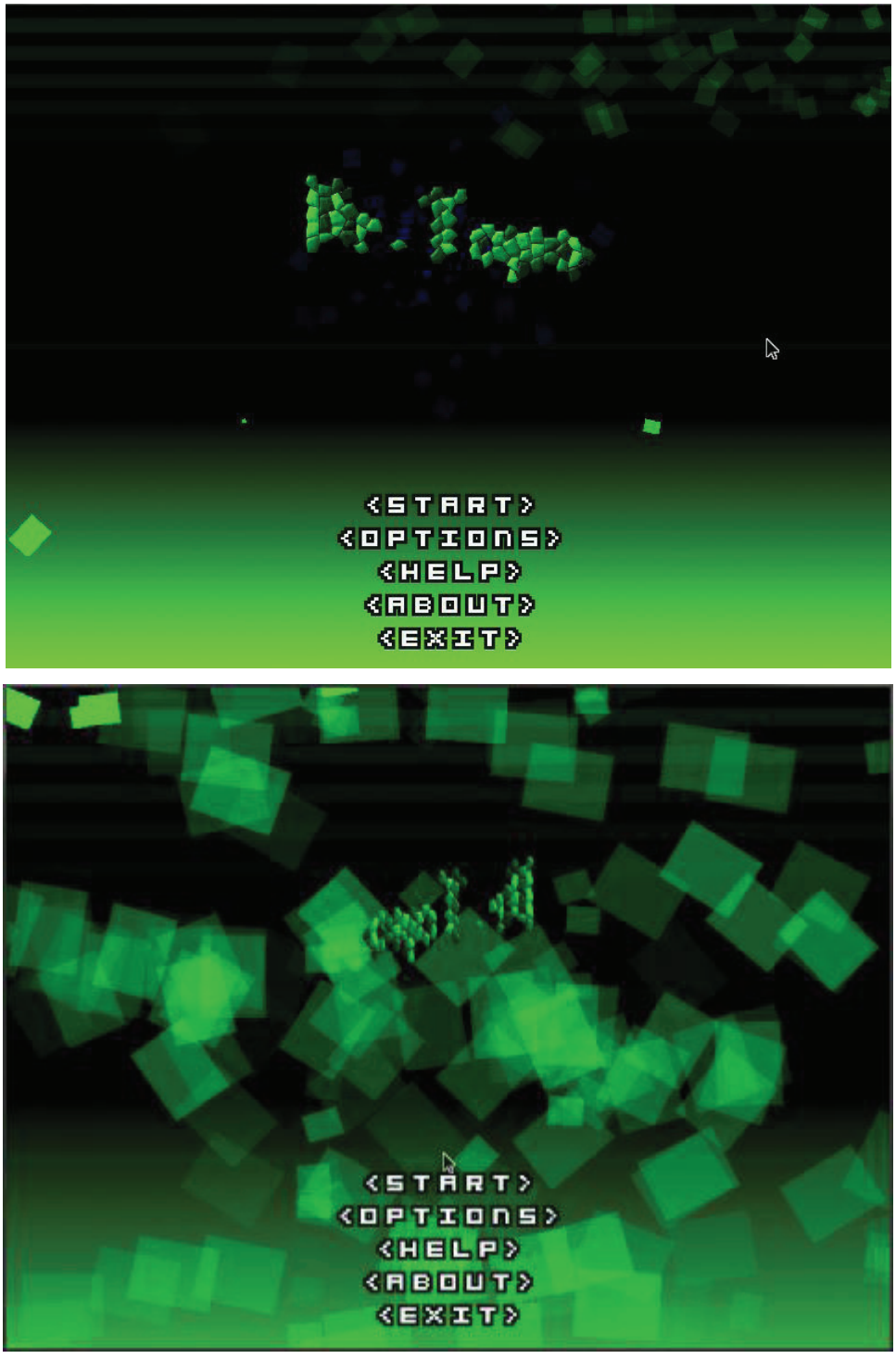
Figura 79. **Gráfica con los resultados de los promedios totales**



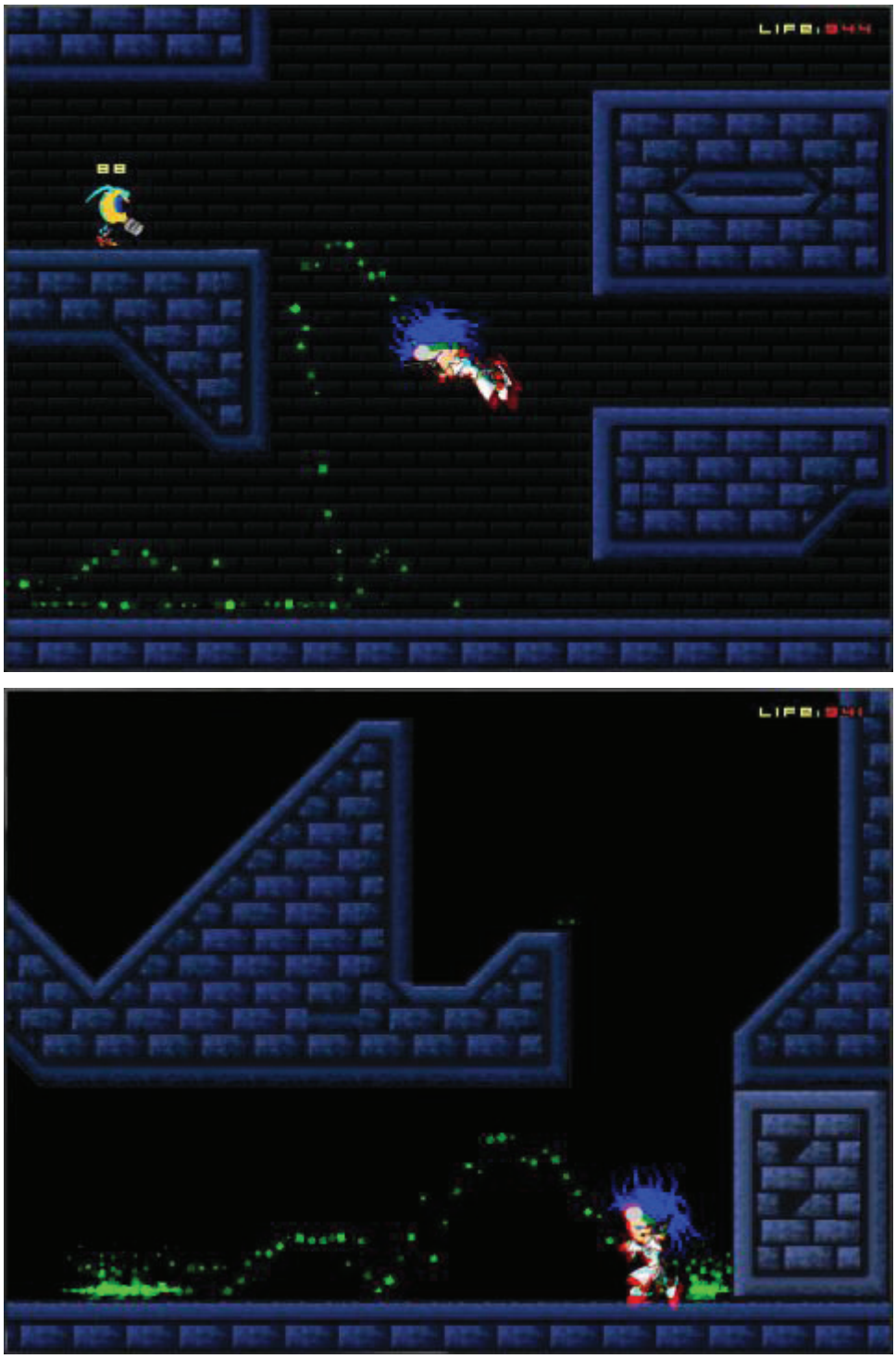
Fuente: propia.

3.4. *Screenshots*

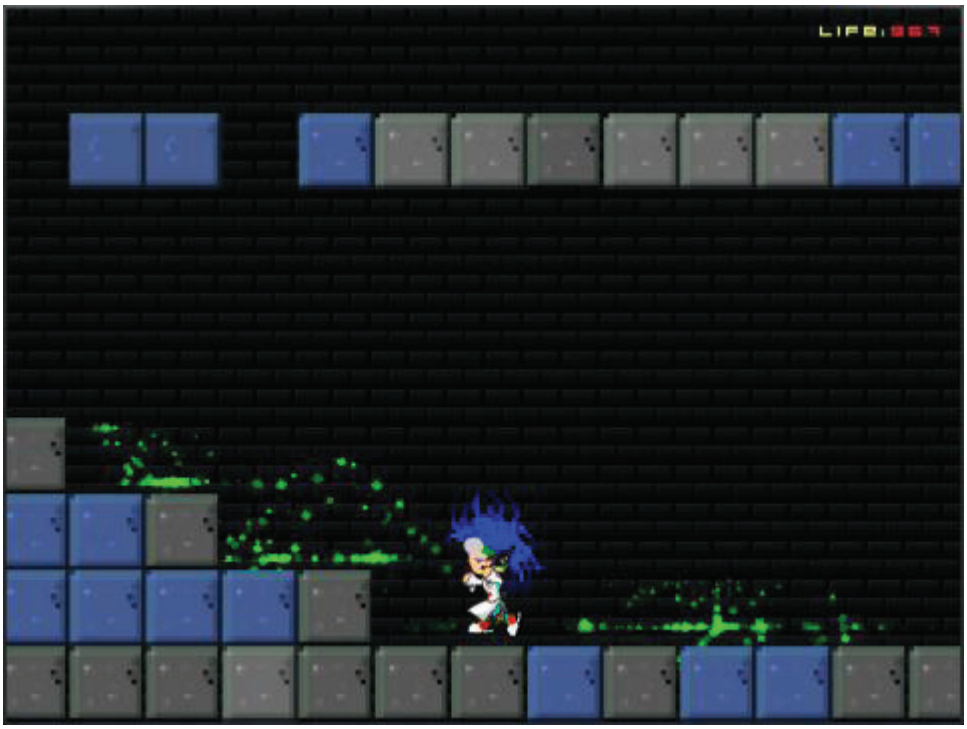
Figura 80. **Imágenes del videojuego publicado**



Continúa figura 80

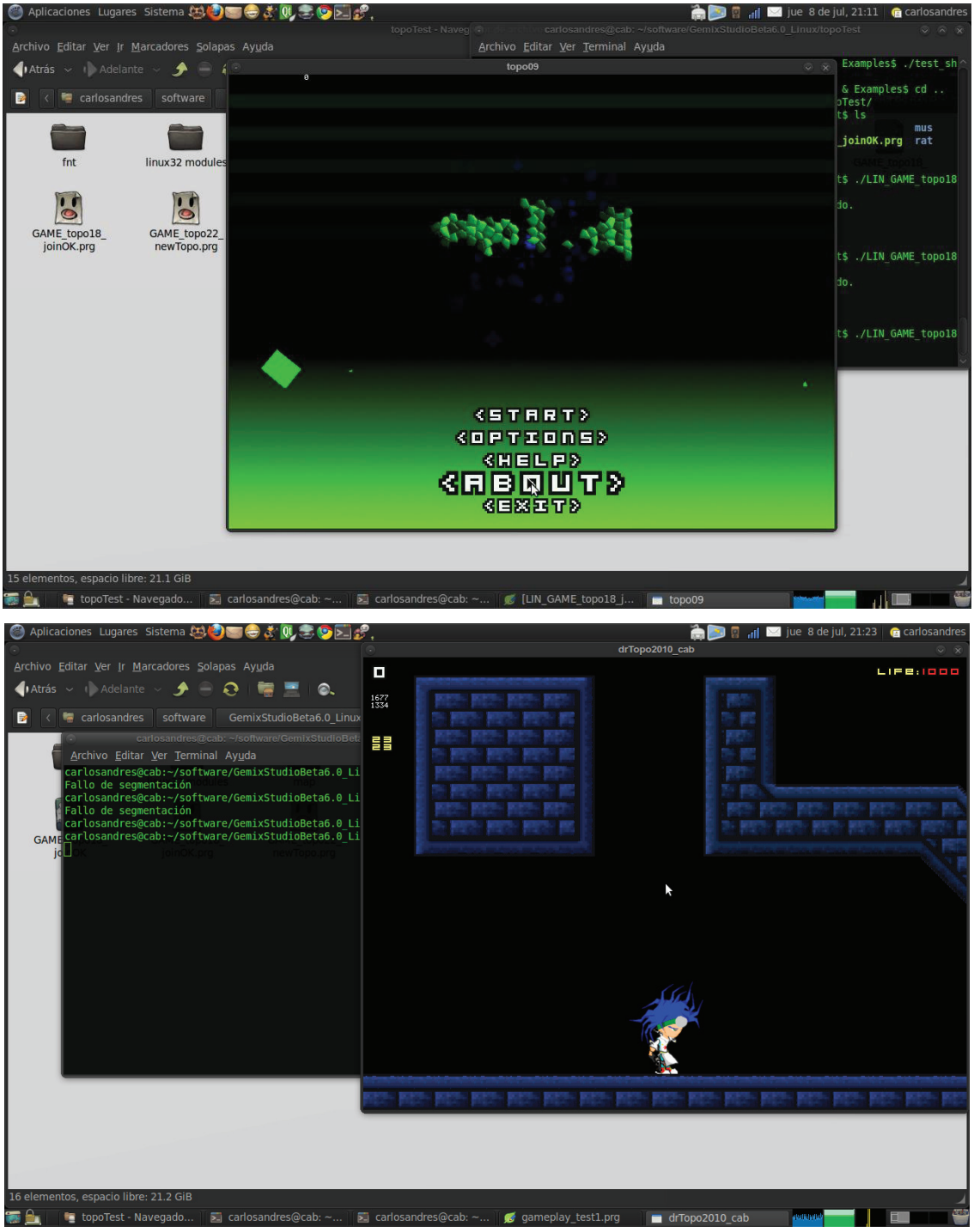


Continúa figura 80



Fuente: propia.

Figura 81. ***Screenshots* del videojuego Dr. Topo en Linux Ubuntu 9.10**



Fuente: propia.

CONCLUSIONES

1. A lo largo de sus más de 30 años de evolución los videojuegos han incorporado las características y capacidades de las nuevas tecnologías como la combinación de varios lenguajes audiovisuales en un mismo soporte, la interactividad, la capacidad para procesar información y conectividad.

2. El proceso de desarrollo de un videojuego es realizado por una sola persona o un grupo de personas que trabajan juntos y pueden ser parte de una gran organización. El desarrollo de un videojuego profesional usualmente incluye las siguientes etapas:

    - Pre-producción
    - Producción
    - Pruebas
    - *Milestones*
    - Mantenimiento

3. La mayoría de juegos comerciales son escritos principalmente en C, C++ y lenguaje ensamblador debido a las complejas limitantes del *hardware* de las consolas. En el caso de los videojuegos para computadora personal se utilizan bibliotecas y APIs como DirectX, OpenGL y SDL. Un lenguaje de programación orientado a procesos es muy útil para la creación de videojuegos debido a que permite programar cada elemento de forma independiente.

4. La selección del lenguaje depende de varios factores como:

    - Lenguajes familiares y conocidos por el equipo de programadores
    - La plataforma de destino
    - La velocidad de ejecución necesaria para el juego
    - El uso de *game engines*
    - Uso de APIs u otras bibliotecas

5. Luego de evaluar las distintas características de los compiladores actuales compatibles con el lenguaje de programación Div, la implementación del videojuego se realizó utilizando Gemix *Studio* debido a su facilidad de uso, soporte técnico, portabilidad, modularidad, estabilidad, efectos gráficos y herramientas incluidas con el compilador.

RECOMENDACIONES

1. Analizar las características, ventajas y desventajas que posee cada herramienta que será utilizada durante el proceso de creación antes de iniciar el proceso de desarrollo. El uso de un compilador multiplataforma permitirá que el videojuego sea portable a distintos sistemas operativos y consolas sin necesidad de reescribir el código del programa.

2. Un lenguaje de programación orientado a procesos permite programar cada elemento del videojuego de forma independiente y sin la necesidad de utilizar bibliotecas o funciones especificas para manejo de múltiples hilos.

3. Para el desarrollo de un proyecto *Indie* los desarrolladores deben documentarse, aprender y conocer el funcionamiento de múltiples herramientas de edición de imágenes, animación, edición de video, edición de audio, etc.

4. Además de los conocimientos técnicos es indispensable desarrollar la creatividad, inventiva e innovación para producir soluciones originales.

REFERENCIAS

1. ASTLE, Dave. *Beginning OpenGL Game Programming*. [S.l.]: Premier Press, 2004. 336 p. ISBN: 1592003699

2. CLINTON, Keith. *Agile Game Development with Scrum.* [S.l.]: Addison-Wesley Signature Series (Cohn), 2010. 312 p. ISBN: 9780321618528

3. FERNÁNDEZ, Pedro. *Librerías genéricas para programación de videojuegos.* Universidad de Oviedo, Escuela Universitaria de Ingeniería Técnica Informática, España, 2002.

4. FORESTE, Tom. *Sociedad de alta tecnología: la historia de la revolución de la tecnología*. [S.l.]: Siglo Veintiuno Editores, S.A., 1992. 346 p. ISBN: 9682316227

5. GIL, A., VIDA, T. *Los Videojuegos*. Barcelona: MEDIAactive, 2007. 128 p. ISBN: 9788497886819

6. GOLDSTEIN, J. *Video Games. A Review of Research*. Informe inédito. Toys Manufacturers of Europe, Bruselas, 1993

7. HARBOUR, Jonathan. *Microsoft Visual Basic game programming with DirectX*. [S.l.]: Premier Press, 2002. 1150 p. ISBN: 193184125X

8. MARTÍN, Ana. *Actividades Lúdicas. El juego, alternativa de ocio para jóvenes*. Madrid: Editorial Popular, 1995. 175 p. ISBN: 8478841555

9. MURPHY, Patrick. *Road To The IGF: World Of Goo's 'Suggested Emotional Journey' To Wii. Gamasutra Videojuegos* [en línea]. Gamasutra [ref. de 20 de diciembre de 2009]. Disponible en Web: <http://www.gamasutra.com/php-bin/news_index.php?story=16749>

10. MYSORE, Sahana. *How the World of Goo became one of the indie video game hits of 2008* [en línea]. Venturebeat. [ref. de 9 de diciembre de 2009]. Disponible en Web: <http://games.venturebeat.com/2009/01/02/the-world-of-goo-became-one-of-the-indie-hits-of-2008/>

11. PLUNKETT, Luke. *Jonathon Blow dice que se gastó 180.000$ en Braid* [en línea]. Kotaku.com. [ref. de 21 de diciembre de 2009]. Disponible en Web: <http://kotaku.com/5037392/jonathan-blow-says-he-spent-180000-on-braid>

12. PROVENZO, Eugene. *Video Kids: making sense of Nintendo.* [S.l.]: Harvard University Press, 1991. 184 p. ISBN: 0674937090

13. RODRÍGUEZ, Elena*. Jóvenes y Videojuegos: Espacio, significación y conflictos*. Madrid: FAD, INJUVE, 2002. ISBN: 9788495248183

14. RODRÍGUEZ, Fernando. *Historia del software de entretenimiento español* [en línea]. Macedonia Magazine [ref. de 25 de diciembre de 2009]. Disponible en Web: <http://macedoniamagazine.frodrig.com/etapa_2001_2002/juegos/Historia%20Soft/soft-5.htm>

15. SÁNCHEZ, Daniel. *Core Techniques and Algorithms in Game Programming*. [S.l.]: New Riders Games. 2003, 888 p. ISBN: 0131020099

16. SANZ, D. *Los videojuegos*. Gaceta Universitaria de la Universidad de Valladolid, 2004, num 474, p. 1-13

17. XBOX.COM. *Braid Xbox Live Arcade* [en línea]. Xbox.com. [ref. de 21 de diciembre de 2009]. Disponible en Web: <http://www.xbox.com/es-ES/games/b/braidxboxlivearcade>

18. 2D BOY. *World of Goo translation for EU release* [en línea]. 2D Boy. [ref. de 20 de diciembre de 2009]. Disponible en Web: <http://2dboy.com/forum/index.php/topic,924.0.html>

BIBLIOGRAFÍA

1. APPLE INC. *Developer Connection* [en línea]. Apple Inc. [ref. de 21 de diciembre de 2009]. Disponible en Web: <http://developer.apple.com/iphone/index.action>

2. BOURG, David. *Physics for Game Developers*. [S.l.]: O'Reilly Media, 2001. 344 p. ISBN: 0596000065

3. BUCKLAND, Mat. *Programming Game AI by Example.* [S.l.]: Jones & Bartlett Publishers, 2004. 495 p. ISBN: 1556220782

4. CEBRIÁN, José. *Fenix: Manual de Referencia* [en línea]. Libro Electrónico, 2000. [ref. de 21 de diciembre de 2009]. Disponible en Web: <http://cloudtdh.googlepages.com/fenix-ref.pdf>

5. CLINTON, Keith. *Agile Game Development with Scrum*. [S.l.]: Addison-Wesley Signature Series (Cohn), 2010. 312 p. ISBN: 9780321618528

6. COHN, Mike. *Scrum for Video Game Development* [en línea]. Mountain Goat Software. [ref. de 1 de julio de 2010]. Disponible en Web: <http://www.mountaingoatsoftware.com/system/presentation/file/50/AgileAustin070206.pdf>

7. DAVISON, Andrew. *Killer game programming in Java*. [S.l.]: O'Reilly Media, Inc., 2005. 969 p. ISBN: 0596007302

8. DEMARIA, Rusel. *High score!: la historia ilustrada de los videojuegos*. Madrid: Osborne McGraw-Hill, 2002. 328 p. ISBN: 9788448137045

9. DÍEZ, Enrique. *La Diferencia Sexual en el Análisis de los Videojuegos*. Madrid: CIDE/Instituto de la Mujer, 2004. 487 p. ISBN: 8468899690

10. EVILDRAGON. *So you want to code in Fenix?* [en línea]. Libro Electrónico. [ref. de 21 de diciembre de 2009]. Disponible en Web: <http://www.gp32x.de/Fenix/Chapter_0.pdf>

11. GEE, James. *Lo que nos enseñan los videojuegos sobre el aprendizaje y el alfabetismo*. [S.l.]: Ediciones Aljibe, 2004. 274 p. ISBN: 8497001680

12. GONZÁLEZ, Roberto. *La Historia de los Videojuegos* [en línea]. Scribd [ref. de 9 de diciembre de 2009]. Disponible en Web: <http://static.scribd.com/docs/5vtrttkhnx7h8.pdf>

13. GOOGLE INC. *Android Developers - Graphics. Android Developers* [en línea]. Google Inc. [ref. de 21 de diciembre de 2009]. Disponible en Web: <http://developer.android.com/guide/topics/graphics/index.html>

14. GROOTJANS, Riemer. *XNA 2.0 game programming recipes: a problem-solution approach*. [S.l.]: Apress, 2008. 875 p. ISBN: 9780321533920

15. HARBOUR, Jonathan. *Beginning game programming*. [S.l.]:Thomson/Course Technology, 2005. 427 p. ISBN: 1598632884

16. LAMOTHE, André. *Tricks of the Windows game programming gurus*. [S.l.]: Sams Publishing, 2002. 1040 p. ISBN: 0672313618

17. LÓPEZ, Daniel. *Análisis Del Contexto Histórico y Tecnológico del Origen de los Videojuegos* [en línea]. Icono14. [ref. de 10 de diciembre de 2009]. Disponible en Web: <http://www.icono14.net/revista/num8/articulos/05.pdf>

18. MICROSOFT MSDN. *The DirectX Software Development Kit SDL* [en línea]. Microsoft Corporation. [ref. de 15 de diciembre de 2009]. Disponible en Web: <http://msdn.microsoft.com/en-us/library/ee416796(VS.85).aspx>

19. NAVARRO, Daniel. *DIV 2 Games Studio*. [S.l.]: Editorial Celesa, 1999. ISBN: 8489245827

20. PALMER, Grant. *Physics for Game Programmers*. [S.l.]: Apress, 2005. 430 p. ISBN: 0131687425

21. PARDO, Jesús. *Breve Historia de los Videojuegos* [en línea]. Ageanet [ref. de 9 de diciembre de 2009]. Disponible en Web: <http://www.agea.org.es/pdf/20060224436/breve-historia-de-los-videojuegos.pdf>

22. SERRANO, Alberto. *Programación de Videojuegos con SDL* [en línea]. Libro Electrónico. [ref. de 10 de diciembre de 2009]. Disponible en Web: <http://www.agserrano.com/libros/sdl/%5Bebook%5DProgramacion%20de%20videojuegos%20con%20SDL.pdf>

23. SYMBIAN OS. *Symbian OS v9.5 product sheet* [en línea]. Symbian. [ref. de 21 de diciembre de 2009]. Disponible en Web: <http://developer.symbian.com/main/downloads/papers/Datasheet_V9.5_final.pdf>

24. TORRENTE, Oscar. *Curso de Iniciación a la Programación de Videojuegos con el Lenguaje Bennu v1.0 (en Windows y GNU/Linux)* [en línea]. Libro Electrónico. [ref. de 25 de febrero de 2010]. Disponible en Web: <http://www.pixjuegos.com/descargas/manualbennu.pdf>

25. TRINIT: Asociación de Informáticos de Zaragoza*. Programación de Videojuegos* [en línea]. TRINIT. [ref. de 20 de diciembre de 2009]. Disponible en Web: <http://trinit.es/temario/>

26. VAN VERTH, James*. Essential Mathematics for Games and Interactive Applications, Second Edition: A Programmer's Guide.* [S.l.]: Morgan Kaufmann, 2008. 704 p. ISBN: 0123742978

27. WALBOURN, Chuck*. Graphics APIs in Windows* [en línea]. Microsoft Corporation. [ref. de 15 de diciembre de 2009]. Disponible en Web: <http://msdn.microsoft.com/en-us/library/ee417756(VS.85).aspx>

28. WIKIPEDIA.ORG. *Historia de los Videojuegos* [en línea]. Wikipedia.org. [ref. de 9 de diciembre de 2009]. Disponible en Web: <http://es.wikipedia.org/wiki/Historia_de_los_videojuego>

29. WOLF, Mark*. The Video Game Theory Reader*. [S.l.]: Francis Group, Inc., 2003. 320 p. ISBN: 0415965799