PROGRAMACION ORIENTADA A OBJETOS

Manual del Programador

Semestre 2016-2017/II

No de proyecto

Nombre del proyecto

Integrantes del equipo

Objetivo del proyecto

Descripción del proyecto

Descripción e imágenes de cada nivel

Diagrama de clases UML

Características y comportamiento de cada clase

Herencia y polimorfismo

Cronograma de actividades (plan de trabajo)

Bitácora de actividades (historial)

No de proyecto

99

Nombre del proyecto

Flappy Eagle

Integrantes del equipo

Jorge Hernán Casillas Cabrera

Objetivo del proyecto

Flappy Eagle es un juego sobre un águila que debe ir volando por el mundo sin caer al suelo, sin chocar con los obstáculos y recolectando comida para así sumar puntos, tendrá 3 tipos de comida y cada una le dará un puntaje diferente al jugador, si cae al suelo o choca con un obstáculo el juego termina, así mismo tendrá 3 tipos de obstáculos diferentes, uno mas difícil de superar que otro, entre mas obstáculos supere y mas comida recolecte su puntaje va a ser mas alto hasta que al fin pierda.

Descripción del proyecto

Un obstáculo es una tubería que aparece arriba y abajo para que el personaje vuele por en medio, al pasar se suma 1 al score, estas aparecen random y si choca contra una de estas, termina el juego. El segundo obstáculo será una planta carnívora saliendo random de ciertas tuberías y al tocarlas se termina el juego. El obstáculo 2 son fantasmas volando por el mundo y al tocarlo igualmente se termina el juego. En cuanto a las comidas, si recolecta unas papas fritas se suma 1 al score, si recolecta la hamburguesa se suma 2 al score y si recolecta la pizza se suman 5 al score y el personaje se vuelve inmune hasta que termine el efecto, entre mas puntaje te de la comida, mas difícil es de recolectar porque se va a mover mas rápido.

Descripción e imágenes de cada nivel

El juego no cuenta con niveles, sin embargo, conforme transcurre el tiempo en el juego, la dificultad aumenta considerablemente, esa es la razón por la que entre más puntos hagamos, nos pueden dar una mejor medalla, la dificultad del juego está en que solo tenemos una vida.

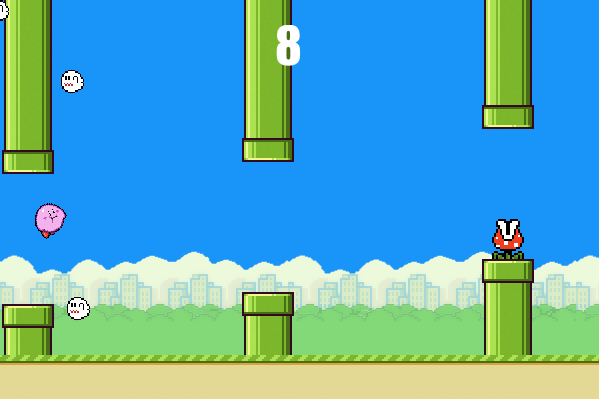
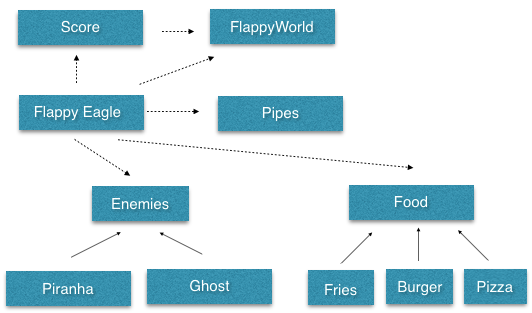


Diagrama de clases UML

Características y comportamiento de cada clase

|  |  |
| --- | --- |
| Nombre de la clase: | FlappyWorld |
| Características: | pipeCounter |
|  | scoreObj |
|  | bgSelection |
| Comportamiento: | createAllPipes() |
|  | spawnGhosts() |
|  | spawnFood() |

* *Para añadir más renglones a la tabla primero debe seleccionar el renglón y después desde el menú “****Tabla****” seleccionar la opción “****Insertar una fila debajo****”*

|  |  |
| --- | --- |
| Nombre de la clase: | FlappyEagle |
| Características: | animFrame |
|  | boolean invencible |
|  | dy |
|  | g (gravedad) |
| Comportamiento: | animate() |
|  | applyGravity() |
|  | checkCollision() |
|  | rotateFlappy() |

|  |  |
| --- | --- |
| Nombre de la clase: | Score |
| Características: | score |
| Comportamiento: | setScore() |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Nombre de la clase: | Pipes |
| Características: | speed |
|  | Bottom |
|  | Top |
| Comportamiento: | removePipes() |

|  |  |
| --- | --- |
| Nombre de la clase: | Piranha |
| Características: | animFrame |
|  | animCounter |
| Comportamiento: | moveLeft() |
|  | animate() |
|  |  |

|  |  |
| --- | --- |
| Nombre de la clase: | Ghost |
| Características: | animFrame |
|  | animCounter |
|  |  |
| Comportamiento: | moveLeft() |
|  | animate() |
|  | getRandomNumber(int start, int end) |

|  |  |
| --- | --- |
| Nombre de la clase: | Food |
| Características: | Clase de padre de todas las comidas. |
|  | xSpeed |
| Comportamiento: | moveLeft(int xSpeed) |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Nombre de la clase: | GameStuff |
| Características: | Clase padre de todos los objetos (imágenes) que simplemente Por ejemplo: el título. |
| Comportamiento: |  |
|  |  |
|  |  |

Herencia y polimorfismo

La herencia en este proyecto fue aplicada simplemente para dar diferentes valores de velocidad a

los objetos, por ejemplo: la clase Food tiene el método moveLeft(int xSpeed), xSpeed es la variable para la velocidad a la que la comida se va a mover, por lo tanto en cada clase comida por ejemplo: Fries, en el método act() simplemente llamo moveLeft() y en el parámetro le doy el valor de velocidad que requiere.

Cronograma de actividades (plan de trabajo)

Describir todas las actividades por realizar desde la propuesta del proyecto hasta la entrega considerando los siguientes entregables:

* + Manual del usuario
  + Manual del programador (este documento)
  + Código
  + Video
  + Link a Greenfoot

|  |  |  |
| --- | --- | --- |
| Fecha de Inicio | Fecha de Término | Actividad por realizar |
| 13/03/2017 | 24/03/2017 | Propuesta de Proyecto |
| 27/03/2017 | 30/04/2017 | Código |
| 01/05/2017 | 15/05/2017 | Manual de Usuario |
| 16/05/2017 | 29/05/2017 | Manual del Programador |
| 30/05/2017 | 30/05/2017 | Link a Greenfoot |
| 30/05/2017 | 30/05/2017 | Video |
| 31/05/2017 | 31/05/2017 | Entrega de Proyecto |

*\* Para añadir más renglones a la tabla debe seleccionar primero la tabla y después desde el menú “****Tabla****” seleccionar la opción “****Insertar una fila debajo****”*

Bitácora de actividades (historial)

Esta parte será llenada durante la elaboración del proyecto. Aquí se deben describir cada una de las actividades realizadas desde la propuesta hasta la entrega del proyecto.

|  |  |
| --- | --- |
| Fecha (dd/mm/aa) | Descripción de la actividad realizada |
| 13/03/2017 | Propuesta de Proyecto |
| 27/03/2017 | Código |
| 01/05/2017 | Manual de Usuario |
| 16/05/2017 | Manual del Programador |
| 30/05/2017 | Link a Greenfoot |
| 30/05/2017 | Video |
| 31/05/2017 | Entrega de Proyecto |

*\* Para añadir más renglones a la tabla debe seleccionar primero la tabla y después desde el menú “****Tabla****” seleccionar la opción “****Insertar una fila debajo***