Realizado por:



Andrea María Rodríguez Amaya, 00366519

Ricardo Javier Beltrán Martínez, 00019315

Xavier Alessandro Quiñonez del Cid, 00048219

Manual técnico

Juego Arkanoid

CONTENIDO

* Aspectos generales.........................................................................2
* Objetivo del documento...........................................................2
* Descripción general..................................................................2
* Software utilizado.....................................................................2
* Modelos utilizados.........................................................................3
* UML Diagrama de clases..........................................................3
* Diagrama relacional normalizado de base de datos utilizada....7
* ER Extendido............................................................................8
* Conceptos técnicos........................................................................9
* Implementación de interfaz gráfica...........................................9
* Manejo de clases en modelo....................................................10
* Posibles errores........................................................................10
* Plataforma base ......................................................................11
* Nomenclaturas ............................................................................12
* Abreviaturas ...........................................................................12
* Eventos y Excepciones................................................................13
* Eventos....................................................................................13
* Excepciones.............................................................................15

Aspectos generales

Objetivo del documento

El objetivo de este documento pretende orientar y explicar el diseño del software creado, explicando las herramientas utilizadas en la creación del juego Arkanoid.

Descripción general

Para la ceración del software se hizo uso del Modelo – Vista – Controlador, o sus siglas en ingles MVC. La aplicación presenta su principal funcionalidad en el entretenimiento del usuario al simular un videojuego arcade donde se muestra la puntuación y el nombre que están registrada en una base de datos.

Software utilizado

Para la creación del programa o aplicación de escritorio se utilizo JetBrains Rider 2020.1.2, en conjunto con PostrgreSQL 12 para la creación de la base de datos de la aplicación. Con los cuales se utilizaron complementos adicionales los cuales fueron Npgsql para la conexión de la base de datos.

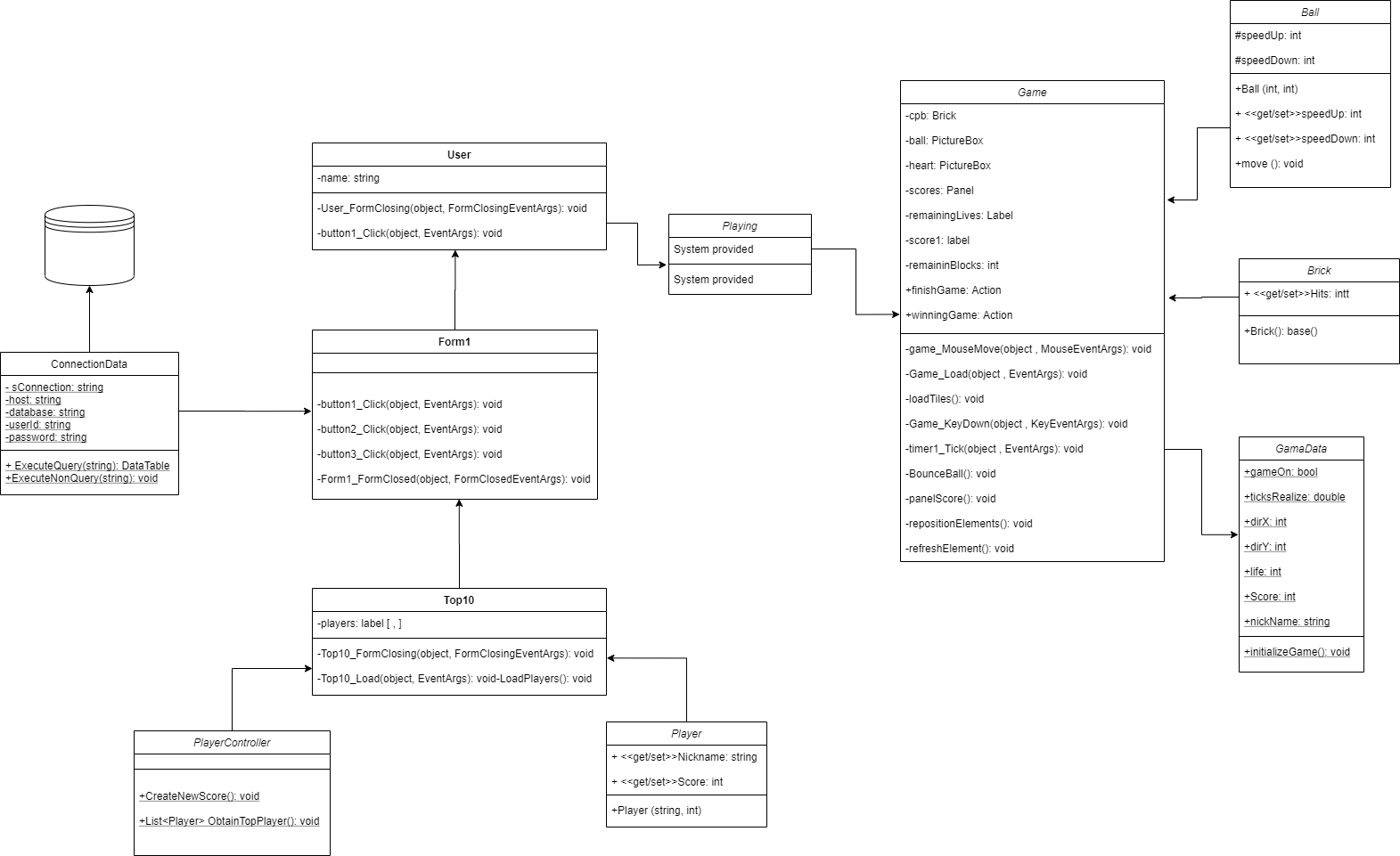
Modelos utilizados

UML Diagrama de clases

El diseño arquitectónico del código está basado en el diagrama de clases siguiente (ver siguiente página):

Se deja un hipervínculo para una vista más detallada de los tres diagramas presentados:

<https://drive.google.com/drive/folders/1h-sDD70kiVO2GXoOs9CEB94naQmflegM?usp=sharing>



Explicando el diagrama de clases, se tiene un usuario por la ejecución de todo la aplicación.

Form1 cuenta unicamente con los diferntes eventos que son los que estan confromados los botones de “Jugar”, “Puntaje”, “Salir” y el botoncito de la “X” en la parte superior del formulario para salir de la palicacion.

La clase Top10 está conformado con dos eventos, el primero que seria cerrar la ventana que lleva de nuevo al usuario al form1 o menu principal y el segundo para cargar el puntaje de los 10 jugadores con la más alta puntuación.

La clase Player tiene un string como Nickname en las cuales pone sus get y set lasd caracterisiticas de los mismos, como asi tambien la variable Score que es de tipo int, con un constructor que permite compartir sus caracteristicas.

La clase PlayerController cuenta unicamente con una lista en donde se almacenan tanto el puntaje del jugador como su nickname. Estos son necesarios en los INSERT de la base de datos local y tambien contiene el CreateNewScore y esta crea un nuevo puntaje por cada vez que inicia un nuevo jugador.

La clase User cuenta con único string que es el name donde el usario ingresa el nickname en un Textbox, luego estan los dos eventos que en este caso seria: El evento que permite presionar la “X” del formulario para cerrarlo y regresar al menu principal. El otro es el boton de comenzar, este perrmite ir al formulario Playing que contiene el UserControl Game, en donde se muestra el juego pero siempre teniendo en cuenta que el usario haya puesto su nickname de lo contrario aparecera una excepción.

La clase Playing es la que permite que el juego empiece a visualizarce e interactuar con el por medio que dentro de ese form se encunetra el usercontrol Game en donde se muestra el juego.

La clase Game (UserControl) contiene 3 diferentes PictureBox (ball, heart) aunque a uno se le llamo Brick y se le puso “cpb” para representar las variables cunado se refierenen a los bloques; tambien consta de un Panel (scores) en donde se mostrara las vidas y puntaje; de dos labels (remainingLives y score1) los cuales muestran las vidas que tiene el usario y su puntaje; un int que seria remainingBlocks donde son los bloques restantes; y terminando con 2 Action (finishGame y winninGame) el primero para cuando termina el juego cuando se terminan las vidas y el otro cuando el jugador destruye todos los bloques. Pero esta clase contiene diferentes eventos y metodos, los cuales son: game\_MouseMove que permite mover la plataforma con el mouse, Game\_load el cual permite donde carga todos los elementos del juego, el metodo loadTiles donde están los elementos. El evento Game\_KeyDown permite que con la tecla “space” se inicie el juego y no presionar otra, y el timer1\_Tick permite que la pelota destruya los bloques, rebote en las paredes de los costados y superior y este cuando la pelota caiga se baje una vida hasta llegar a 0 y finalizar el juego. El método BounceBall contiene las físicas de la pelota para realizar los rebotes, el metodo panelScore muestra los elementos de la vida y del puntaje, el método repositionElements, para que la plataforma y la pleota vuelva su pocisión original al caer la pelota. El método refreshElements actualiza y muestra la vida correspondiente cuando la pelota cae y se pierde una vida.

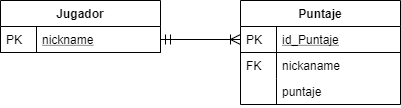
En la clase Ball se encuentran las caracteristicas de la pelota, la velocidad de ella cuando se dirige hacia arriba y hacia abajo cunado rebota. Su contructuctor permite compartir sus caracteristicas con otras clases, tambien con un método que hace mover a la pelota en la clase Game.

La clase Brick contiene las caracteristicas de tipo int de la variable Hits para que los bloques se destruyan dependiendo de los golpes que le den la pelota.

La clase GameData contiene una bandera, o un booleano, para que ésta detecte cuando se inicia el juego por medio de la tecla “space”. Las variables dirX y dirY de tipo int realizan la funcion de las direcciones en la que se dirige la pelota. La variable life (vida), de tipo int, muestra el total de vidas que tendrá el jugador al inicio del juego. La variable score (puntaje) establece que el puntaje del jugador debe de empezar en un total de 0. Por último, la variable nickName, de tipo string, almacena el nombre del jugador. El método initializeGme hace la función para que el juego inicie.

Diagrama relacional de base de datos utilizada

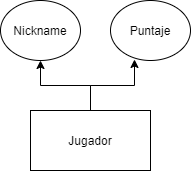
Complementariamente, el esquema brindado de la base de datos se segmenta de la siguiente manera:



El previo esquema permite relacionar que cuando el usuario escriba su nickname éste se ingrese a la base de datos y cuando el jugador empiece a jugar y éste almacene puntos al romper los bloques, a su vez se ingresan también en la base de datos junto al nickname del usuario. Cuando se selecciona la opción “Puntaje” del menú principal, éste muestra los primeros 10 jugadores con el puntaje mas alto que esta almacenada en la base de datos.

Diagrama ER Extendido

En el previo diagrama se muestra como la entidad jugador se relaciona con los atributos llamados Nickname y puntaje, en la que se relacionan en una base de datos.



Conceptos técnicos

Implementación de interfaz grafica

La interfaz gráfica de la aplicación consiste en cuatro formularios, compuestos de distintos botones, caja de texto (TextBox en inglés). En el formulario llamado “Playing.cs” se encuentra un control de usuario (UserControl en inglés) llamado “Game.cs”. En este control de usuario el jugador o usuario puede empezar a jugar el juego y, a su vez, podrá observar lo que contiene el mismo, por ejemplo: etiquetas (labels en inglés), caja de imágenes (PictureBox en inglés), los bloques, las vidas, el puntaje, la pelota son programados para que el funcionamiento que contiene tanto la física de la pelota como del puntaje y de las vidas sean más fáciles; de manejar que, cuando las gráficas están en el diseñador (Designer en inglés), los controles de usuario y formularios implementados son:

**Formulario:**

* Form1.cs
* Playing.cs
* Top10.cs
* User.cs

**Control de usuario:**

* Game.cs

Manejo de clases en modelo

Para manejar la parte fundamental del modelo de la aplicación, se contaron con las diferentes clases:

**Clases:**

* GameData.cs
* Player.cs
* PlayerController.cs
* Ball.cs
* Brick.cs
* ConnectionDB.cs
* Program.cs

Posibles errores

Dentro de los posibles errores que el programa podría llegar a presentar se encuentran el lag, tanto de la plataforma como del fondo del juego, y un pequeño bug por parte de la pelota y la plataforma.

Al moverse la plataforma, el fondo presenta un poco de distorsión alrededor de la plataforma. Es un lag casi imperceptible, pero está presente. Esto ocurre ya que ambas imágenes, tanto de la plataforma como el fondo, utilizan muchos recursos por parte de la máquina y el IDE. Para mejorar este aspecto visual se deben manejar recursos internos de la plataforma de código. Sin embargo, esto supone un manejo de los recursos internos del IDE y el lenguaje de programación. Hacer manejo de estos recursos vuelve más complicado la creación del juego y la programación de las gráficas.

Otro error que se puede percibir es un bug cuando la plataforma toca el lado lateral derecho. Aunque esto no siempre ocurre, puede darse el caso que la plataforma no pueda tocar el lado latera derecho, como si algo se lo impidiese, pero al volver a intentarlo la plataforma sí llegara al límite del lado derecho. Como el lag del fondo y la plataforma, esto ocurre por la cantidad de recursos que requiere de la computadora y el IDE.

Con respecto a la pelota y la plataforma, si la plataforma no llegase a tocar el límite del lado lateral derecho y la pelota fuera a caer de ese lado, la pelota no caerá al vacío causando que el jugador pierda una vida. En este caso, la pelota pasará por debajo de la plataforma, rodeándola y regresando a la parte superior, sin causar una pérdida de vida, y seguirá como si hubiese rebotado en la plataforma.

Plataforma base

|  |  |
| --- | --- |
| Sistema operativo | Multiplataforma |
| Tecnologías | **JetBrains Rider 2020.1.2** |
| Lenguaje | **C#** |
| Gestar de DB | **PostrgreSQL 12** |

Nomenclaturas

Abreviaturas

Para los diferentes elementos que se usaron en el entorno grafico se diseño una tabla para demostrar su abreviatura, ya que se uso abreviaturas al usar los elementos y así facilitar el uso de ellos.

Las abreviaturas se presentan a continuación:

|  |  |
| --- | --- |
| Label | Lbl |
| TextBox | **Txt** |
| PictureBox | **cpb, pb** |
| Button | **Bttn** |
| Panel | **pnl** |
| TableLayoutPanel | **tlp** |

Eventos y excepciones

Eventos

Para poder pasar a una acción reflejada en control de usuario o en un formulario fue necesario la implementación de eventos. Los eventos implementados fueron:

**Form.cs**

* button1\_Click (object sender, EventArgs e)

Muestra el formulario donde el usuario podrá ingresar su nickname.

* button2\_Click (object sender, EventArgs e)

muestra el formulario donde el usuario puede ver los 10 puntajes mas altos guardados en la base de datos de las personas que han jugado Arkanoid.

* button3\_Click (object sender, EventArgs e)

Realiza la función de finalizar la aplicación.

* Form1\_FormClosed (object sender, FormClosedEventArgs e)

Realiza la función de salir del juego dando click en la “X” del formulario.

**User.cs**

* User\_FormClosing (object sender, FormClosingEventArgs e)

Realiza la función de cerrar el formulario de registrar el usuario para así poder regresar al formulario principal en donde se muestra el menú principal.

* button1\_Click (object sender, EventArgs e)

Este botón hace que después que el usuario haya ingresado su nickname en el TextBox este pueda ir a un nuevo formulario en donde se encuentra el UserControl que es el juego Arkaoid.

**Top10.cs**

* Top10\_FormClosing (object sender, FormClosingEventArgs e)

Cierra el formulario del Top 10 para que el usuario vuelva al menú principal.

* Top10\_Load (object sender, EventArgs e)

Muestra a los 10 jugadores con el puntaje mas alto que aparece en la base de datos.

**Game.cs**

* game\_MouseMove (object sender, MouseEventArgs e)

Este permite que el usuario pueda controlar mediante el ratón (Mouse en inglés) la plataforma de izquierda a derecha.

* Game\_Load (object sender, EventArgs e)

Acá se permite que se cargue donde es la plataforma, la pelota, los bloques, las vidas, el puntaje, que se reinicien estos cuando se cambie de jugador, también se define el ancho, alto y su posición en el UserControl.

* Game\_KeyDown (object sender, KeyEventArgs e)

Se hace una verificación para que el usuario tenga que presionar la tecla correcta que en este caso seria la tecla “space” o barra espaciadora y si se presiona otra tecla manda un mensaje que debe oprimir la tecla correcta.

* timer1\_Tick (object sender, EventArgs e)

Permite que la pelota rebote en las paredes de los costados , en los bloques, en la pared superior, en la plataforma y que cuando este sobrepase el desnivel de la plataforma se pierda una vida y que con la misma la pelota regrese a la plataforma como en el inicio del juego pero con menos vida y así sucesivamente hasta que el marcador de la vida llegue a 0 el cual mandara un mensaje diciendo que se ha perdido y que finalizara la aplicación y regresara al menú principal.

Excepciones

Las excepciones, debido a sus nombres, son auto-explicativas. Ya que constan de un instructor que recibe un string con el mensaje de un error. La aplicación consta de las siguientes excepciones:

**Excepciones:**

* EmptyNicknameException.cs

Esta excepción surge si el jugador no ingresa un nombre al registrarse o ingresa solo espacios. Se le impedirá el acceso al juego a menos que ingrese un nombre.

* ExistingNicknameException.cs

Esta excepción ocurre si el usuario ingresa un nombre ya existente en la base de datos. Se le pedirá que ingrese un nombre que no esté almacenado en la base de datos para poder acceder al juego.

* LongerNicknameException.cs

Esta excepción surge si el nombre que ingreso el usuario tiene más de 15 caracteres. Dado que, la base de datos está programada de manera que el nombre sea una variable VARCHAR con límite de 15 caracteres, se controla como excepción en el código.

* NoRemainingLivesException.cs

Esta excepción ocurre cuando el jugador pierde sus tres vidas. Cuando el conteo de vidas llega a cero, entonces se lanzará la excepción y regresará al menú principal.

* OutOfBoundsException.cs

Esta excepción surge cuando la pelota sale de los límites de la pantalla, es decir, cuando el jugador deja caer la pelota. Si llegase a caer y el jugador se queda sin vidas, entonces lanza la excepción NoRemainingLivesException en conjunto.

* WrongKeyPressedException.cs

Esta excepción ocurre cuando el usuario presiona cualquier tecla que no sea “space” para iniciar el juego. Al ocurrir esta excepción, se le impedirá al jugador iniciar el juego y se le pedirá que presione “space”.