MANUAL TECNICO

Felidae Arkanoid

Exception

Realizado por:

Natalia Alexandra Solórzano Paz 00120319

Carmen Elisa López Alvarado 00100619

José Heriberto Olivares Barrientos 00177919

Contenido:

Aspectos Generales:

Objetivo del documento:

El objetivo de este manual técnico tiene como objetivo explicar y motivar a otros programadores a realizar un juego "Arkanoid" personalizado.

Descripción general:

El proyecto consiste en la eliminación de bloques por medio de una bala, el jugador puede pasar de nivel cada vez que destruya toda la muralla. Se ha utilizado el modelo – vista – controlador MVC, que es un patrón de desarrollo de sistemas.

Software utilizado:

Para la creación del programa se utilizó JetBrains.Rider-2019.3.4, en conjunto con PostgreSQL 11, el DBMS, para crear la base de datos. Uso de Drawio para la creación de diagramas.

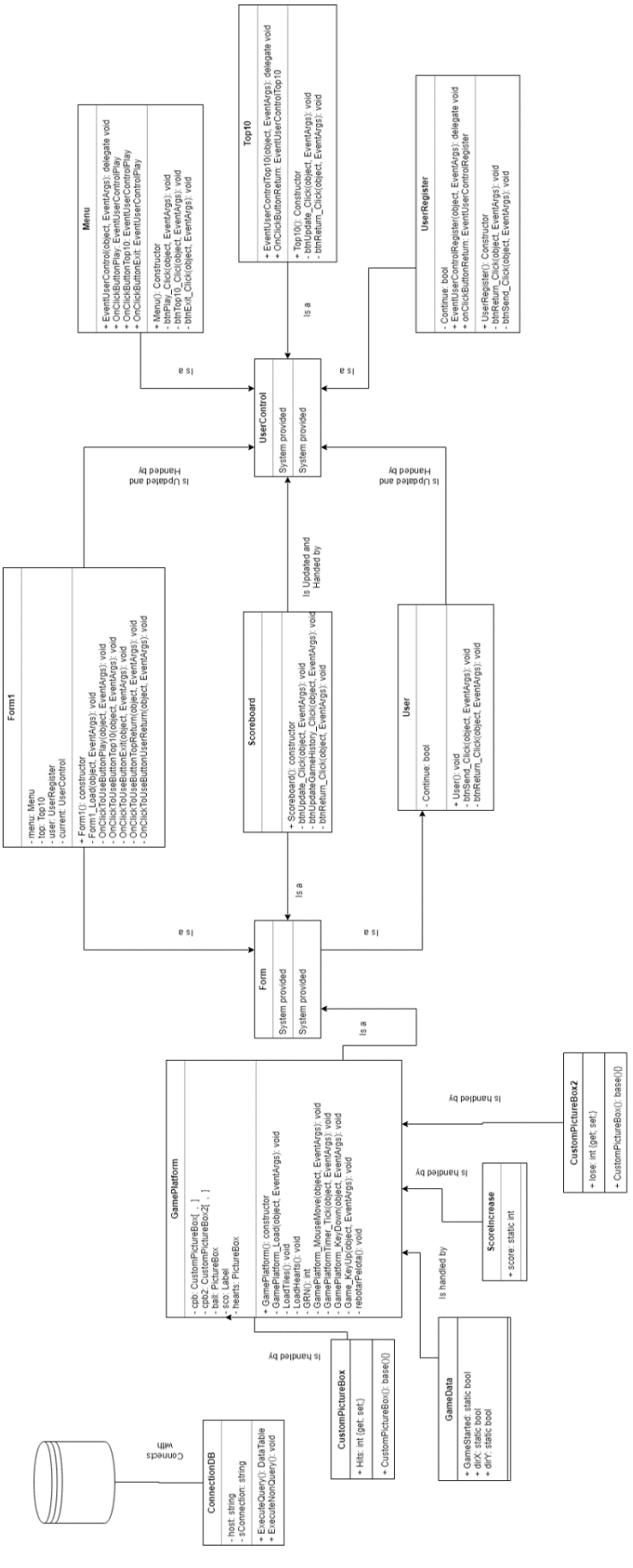
Modelos Utilizados:

UML Diagrama de Clases:

El diseño arquitectónico del código esta establecido en el diagrama de clases siguiente:

Se deja un hipervínculo para una vista más detallada:

file:///C:/Users/Nathy/Documents/Ciclo%20I%202020/Programacion%20Orientada%20a%20Objetos/Proyecto%20Final/UMLFelidae.html

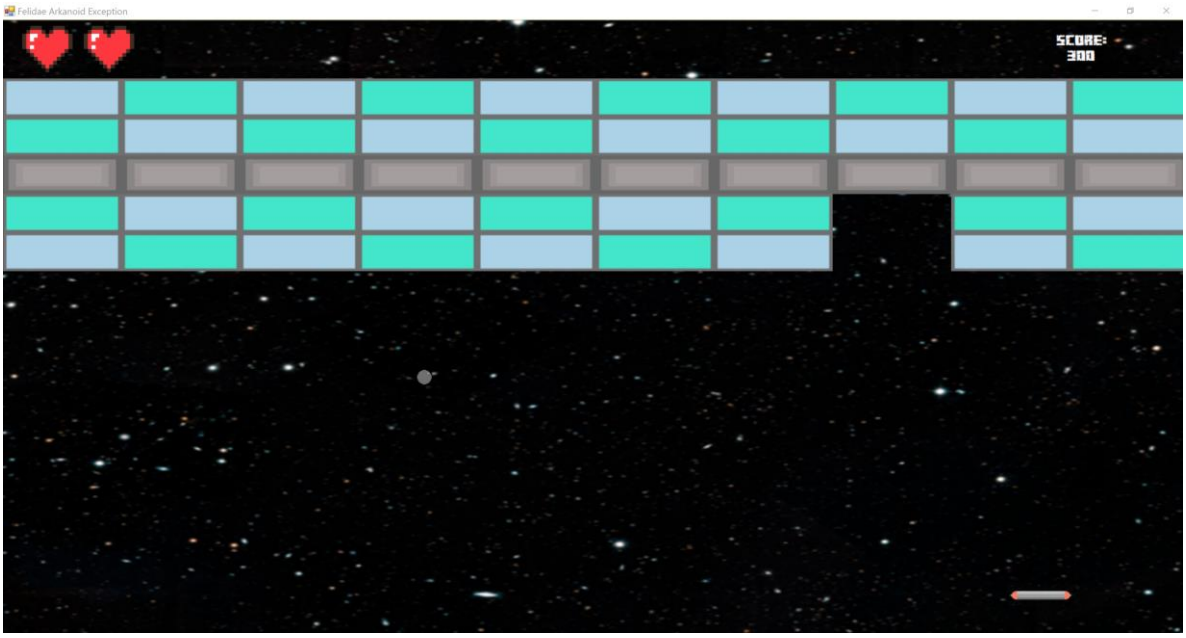

Menu
+ EventUserControl(object, EventArgs); delegate void
+ OnClickButtonPlay: EventUserControlPlay
+ OnClickButtonTop10: EventUserControlPlay
+ OnClickButtonExit: EventUserControlPlay
+ Menu(); Constructor
- btnPlay_Click(object, EventArgs): void
- btnTop10_Click(object, EventArgs): void
- btnExit_Click(object, EventArgs): void
Top10
+ EventUserControlTop10(object, EventArgs); delegate void
+ OnClickButtonReturn: EventUserControlTop10
+ Top10(); Constructor
- btnUpdate_Click(object, EventArgs): void
- btnReturn_Click(object, EventArgs): void
UserRegister
- Continue: bool
+ EventUserControlRegister(object, EventArgs); delegate void
+ onClickButtonReturn: EventUserControlRegister
+ UserRegister(): Constructor
- btnReturn_Click(object, EventArgs): void
- btnSend_Click(object, EventArgs): void
Is a
Is a
Is a
UserControl
System provided
System provided
Is Updated and Handled by
Is Updated and Handled by
Form1
- menu: Menu
- top: Top10
- user: UserRegister
- current: UserControl
+ Form1(): constructor
- Form1_Load(object, EventArgs): void
- OnClickToUseButtonPlay(object, EventArgs): void
- OnClickToUseButtonTop10(object, EventArgs): void
- OnClickToUseButtonExit(object, EventArgs): void
- OnClickToUseButtonTopReturn(object, EventArgs): void
- OnClickToUseButtonUserReturn(object, EventArgs): void
Scoreboard
+ Scoreboard(): constructor
- btnUpdate_Click(object, EventArgs): void
- btnUpdateGameHistory_Click(object, EventArgs): void
- btnReturn_Click(object, EventArgs): void
User
- Continue: bool
+ User(): void
- btnSend_Click(object, EventArgs): void
- btnReturn_Click(object, EventArgs): void
Is a
Is a
Is a
Form
System provided
System provided
Is a
GamePlatform
- cpb: CustomPictureBox[,]
- cpb2: CustomPictureBox2[,]
- ball: PictureBox
- score: label
- hearts: PictureBox
+ GamePlatform(): constructor
+ GamePlatform_Load(object, EventArgs): void
- LoadTiles(): void
- LoadHearts(): void
- GRN(): int
- GamePlatform_MouseMove(object, EventArgs): void
- GamePlatformTimer_Tick(object, EventArgs): void
- GamePlatform_KeyDown(object, EventArgs): void
- Game_Key Up(object, EventArgs): void
- rebotarPelota(): void
Is handled by
Is handled by
Is handled by
Is handled by
CustomPictureBox2
+ lose: int {get; set;}
+ CustomPictureBox2(): base()()
ScoreIncrease
+ score: static int
CustomPictureBox
+ Hits: int {get; set;}
+ CustomPictureBox(): base()()
GameData
+ GameStarted: static bool
+ dirX: static bool
+ dirY: static bool
Is handled by
Connects with
ConnectionDB
- host: string
- sConnection: string
+ ExecuteQuery(): DataTable
+ ExecuteNonQuery(): void

Habrá un menú principal en un “UserControl” llamado “Menu”, que se presenta en un “Windows Form” llamado “Form1” que contiene las opciones siguientes:

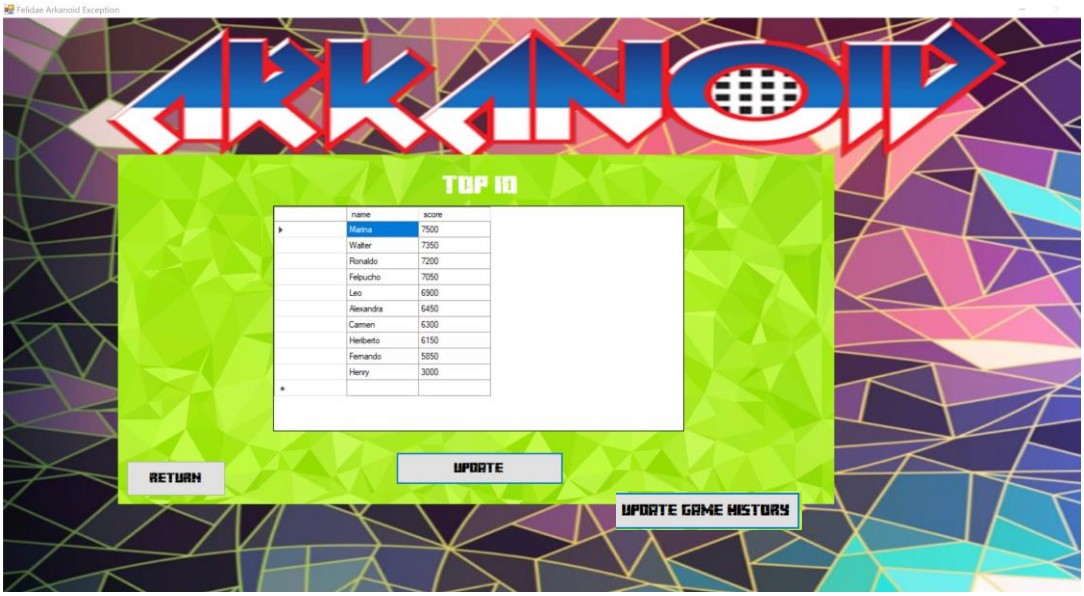
Habrá un registro de usuario en un “UserControl” llamado “UserRegister”, que se presenta en un “Windows Form” llamado “User”, en donde se conecta a la base de datos para guardar el nombre de usuario que jugará en ese momento “Felidae Arkanoid Exception”.

Existe otro “Windows Form” llamado “GamePlatform”, que consiste en un método de eliminación de bloques de colores, llamada LoadTiles(object, EventArgs), se deberán ir eliminando con una bala, usara un metodo para que la bala pueda seguir avanzando llamado rebotarPelota().

La bala estará en un pedestal dinámico que permita cambiar la dirección de la bala en cada toque, que también permitirá una velocidad y un inicio de juego utilizando los metodos GamePlatformTimer_Tick(object, EventArgs) y el GamePlatform_MouseMove(object, EventArgs) para que el pedestal se mueva junto a los toques que se hacen el “mousePad” si el usuario logra terminar todo el juego, puede avanzar y seguir jugando, si no logra terminar todo el juego pierde. Los puntajes se irán guardando en la base de datos en la tabla “scoreboard”. Tambien habrá una función que brinda 3 vidas principales a cada jugador, serán manipuladas desde el metodo LoadHeart().



Por último, existirá una comparación de puntajes entre los jugadores, se usará un “UserControl” llamado “Top10”, presentado en un “Windows Form” llamado “Scoreboard”, que permitirá visualizar los puntajes de todos los jugadores. Obteniendo los datos de las tablas de la base de datos “player” y “scoreboard”. Mostrando el nombre del jugador y el puntaje que obtuvo.



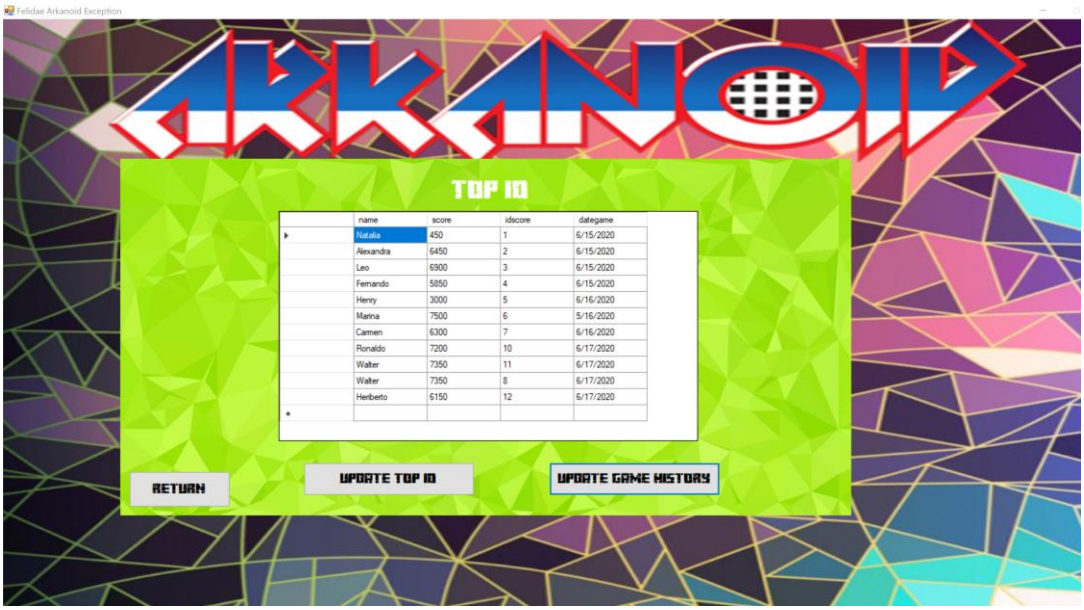
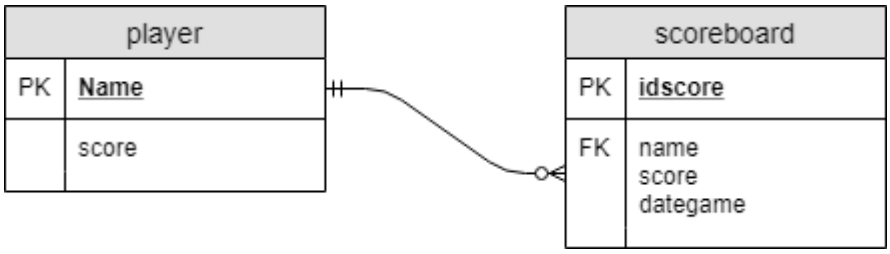


Diagrama Relacional Normalizado de base de datos utilizada:

La siguiente imagen muestra el primer Diagrama Relacional que realizamos, no permite relacionar al jugador con la tabla de puntajes.

La tabla “player”, contiene el nombre “Name” de los jugadores, y el “score” del puntaje de cada el jugador con respecto a los otros jugadores. Mostrando el Top 10.

La tabla “scoreboard”, contiene el “idscore” que es serial que muestra el número de partidas, muestra el “name” y el “score” de cada jugador, junto a la fecha en que se realizo la jugada.



Conceptos Técnicos:

Implementación de interfaz gráfica:

La interfaz gráfica del programa es compuesta de cuatro "Windows Forms", incluyendo buttons, pictureBox, Labels, Paneles Principales (Game_Load, para el form "Game".) Cada ventana, provee distintas opciones de la solución, y son los siguientes:

- Form1.cs
- User.cs
- GamePlatform.cs
- Scoreboard.cs
- Menu.cs
- Top10.cs
- UserRegister

Manejo de clases en modelo:

Para el manejo de clases solamente se utilizó una clase estática para conectar el juego a la base de datos:

- ConnectionDB.cs

Plataforma base:

Sistema Operativo	Multiplataforma
Tecnologías	JetBrains Rider 2019.3.4
Lenguaje	C#
Gestor de DB	PostgreSQL
Diseñador de Diagramas	Draw.io Diagrams

Nomenclaturas:

Abreviaturas:

La nomenclatura que se ha utilizado a lo largo del código ha sido la siguiente abreviatura regulada:

<Abreviatura de tipo>_descripción

Las abreviaturas gestionadas son:

Label	lbl
PictureBox	pic
Button	btn
DataGridView	dgv
TextBox	txt

Eventos y Excepciones:

Eventos:

- EventUserControl.cs

En este evento se muestra un menú principal, que contiene tres opciones: “Play”, “Scoreboard”, “Exit”, dependiendo a que botón se seleccione, registrara un click en “UserRegister.cs”, para registrarse; registrara un click en “Top10.cs”, para ver todos los puntajes de los jugadores; Y finalmente se puede registrar un click en “Exit”, que permite salir del juego.

- EventUserControlRegister.cs

Este evento permite registrar un nuevo jugador, y envía los datos a la base de datos para guardarlos, con el botón “Send”. También permite retornar al menú principal con un botón “Return”.

- EventUserControlTop10.cs

Este evento permite comparar y observar el “Top 10”, de todos los jugadores que ya han realizado un registro y una jugada. Permite con el botón “Return”, regresar a la pantalla de menú; También aparece una opción “Update”, para mostrar cualquier cambio que haya pasado desde la última jugada.

Excepciones:

- ExceptionNoInfo

Es para evitar que queden espacios de texto vacíos.

- ExceptionExit

Evita que haya errores al querer salir del juego.

- ExceptionUpdateScores

Muestra los datos correctos del Top 10.

- ExceptionGame

Permite que el juego fluya sin interrupción.

- ExceptionGameOver

Da la opción al jugador de volver a jugar o regresar al menú principal.

- ExceptionMouseMovement

Evita que se desconecte el movimiento del mouse con la tabla y la pelota

- ExceptionGameTimer

Permite que cuando el juego comience no haya ningún error

- ExceptionBallBounds

Ayuda a que la pelota se mantenga dentro de la plataforma del juego