Technical Manual

For

GradedUnit Game

Ver. 1

22/04/15

Written by Dylan Gilmour

Contents

[Project Brief 3](#_Toc417595963)

[Requirements 3](#_Toc417595964)

[Hardware Requirements 3](#_Toc417595965)

[Software Requirements 3](#_Toc417595966)

[Functionality 4](#_Toc417595967)

[Functional Comments 4](#_Toc417595968)

[Non-Functional Comments 4](#_Toc417595969)

[Additional Functionality 4](#_Toc417595970)

[Class Design 5](#_Toc417595971)

[Class Diagram 6](#_Toc417595972)

[External Libraries 6](#_Toc417595973)

[Database Design 7](#_Toc417595974)

[ERD 7](#_Toc417595975)

[Player 7](#_Toc417595976)

[High Scores 7](#_Toc417595977)

[Mode 7](#_Toc417595978)

[SQL Commands 8](#_Toc417595979)

[Testing 8](#_Toc417595980)

[Methodology 8](#_Toc417595981)

[Testing Plan 8](#_Toc417595982)

[Test Cases 8](#_Toc417595983)

[Appendix 8](#_Toc417595984)

[Data Dictionary 8](#_Toc417595985)

[Code 8](#_Toc417595986)

# Project Brief

For my project as part of my HND Software Development I will be creating a local multiplayer game in which multiple users cooperate in clearing waves of ‘bricks’ by controlling a ‘bat’ in order to bounce a ‘ball’ around the screen in a similar style to the classic “Breakout” which was released in 1972 by Atari Inc.

I have chosen C# for the development language. This is because C# is known for its versatility and portability, and is a commonly used language in game development.Requirements

**As the game will be operated via keyboard all graphical elements should be easy to control with the keyboard. The game has a very simplistic design and is light on computing resources and should be able to run on any modern pc with the correct graphics specification. A keyboard will be required to interact with the game in anyway.**

## Hardware Requirements

|  |  |  |
| --- | --- | --- |
|  | Minimum | Recommended |
| Operating System: | Windows XP with SP2 | Windows 7 / 8(.1) |
| Processor: | 1GHz |  |
| Graphics Card: | Support for Shader Model 1.1 and DirectX9.0C |  |
| Memory: | 512MB |  |
| Sound Card: | N/A |  |
| Input: | Keyboard | Keyboard |

## Software Requirements

**Any software required**

Systems wishing to run GradedUnit Gamewill be required to have .Net Framework 4 installed.

As this game is built upon the XNA framework it will require the XNA redistributable to be installed on the target machine. In order for the database to properly function or for the user to manually edit it the Microsoft access runtime may be required this can be found in the link provided.

Microsoft access runtime: http://www.microsoft.com/en-gb/download/details.aspx?id=39358

## Functionality

|  |  |
| --- | --- |
| Functional Requirements | |
| Requirement | **IMPLEMENTED?** |
| Players will be presented with a menu system to choose which mode the game will run in | Yes |
| Players will be able to rebind the control system in a separate menu which can be accessed when the game is run | No |
| Players will have a method of control which will allow the bats to be moved horizontally along the screen and release the ball when it gets reset | Yes |
| The Menu system will be able to be controlled by the keyboard and the mouse | Keyboard functionality yes mouse no |
| Players will have 4 lives each to start with | Yes |
| High scores will be stored in a file which will be updated after each game over | Yes |
| High scores will be loaded at the high scores screen | Yes |
| High scores will be able to be viewed by a separate option in the main menu | Yes |
| Upon the ball making contact with the wall, bat or bricks the ball will ‘bounce’ and change its velocity | Yes |
| Player scores will be displayed on the game screen when the game is active | Yes |
| Player lives will be displayed when the game is active | yes |

### Functional Comments

**Due to time constraints the availability to rebind the controls will not be available in this release of the game. As the entire game will be controlled via keyboard I chose not to implement mouse**

|  |  |
| --- | --- |
| non-Functional Requirements | |
| Requirement | **IMPLEMENTED?** |
| The bats will initially be coloured red and blue | Yes |
| The blocks will be red, blue, green, and orange | Yes |
| Colour of the bats will be able to change in the settings menu | No |
| Systems wishing to run the game will require to have .Net Framework 4 to run the games created | yes |
| Systems will require Windows XP or higher but primary focus will be on Windows 7 and Windows 8(.1) | yes |
| Hardware requirements are a graphics card that can support a minimum of Shader Model 1.1 and DirectX 9.0c in accordance with the requirements of XNA framework | yes |
| The game backdrop will be black in colour | yes |

### Non-Functional Comments

**Due to time constraints the changing of colour will not be available in this release**

### Additional Functionality

**Added in an extra row of bricks with colour Pink to fill up extra screen space**

# Class Design

|  |  |  |
| --- | --- | --- |
| Class Name | BAll | |
| UsAGE | Holds the values and methods for the Ball | |
| Initial Design | ball | |
| Changes from Initial Design | Changed most variable types to utilise xna framework values | |
| Variables | **NAME** | **USAGE** |
| Boundary  Collision  Initialspeed  Motion  Pos  Random  scrBoundary  speed  texture | Holds a rectangle for the boundary of the ball  Holds whether or not the ball has collided  Holds the initial speed of the ball  Holds the motion of the ball  Holds the position of the ball  Holds random number  Holds the boundary for the screen  Holds the speed of the ball  Holds the texture of the ball |
| Methods | **NAME** | **USAGE** |
| Ball  BatCollision  BottomCheck  TopCheck  Deflection  Draw  getMotion  StartMotion  StartPosBall  UpdatePos | Constructor  Checks if ball has collided with a bat  Checks if the ball has went off the top of the screen  Checks if the ball has went off the bottom of the screen  Changes the velocity of the ball when it hits something  Draws the ball  Returns the motion of the ball  Starts the ball moving  Set the starting position of the ball  Updates the position of the ball |

|  |  |  |
| --- | --- | --- |
| Class Name | Bat | |
| UsAGE | Used to hold and move the player controlled objects | |
| Initial Design | bat | |
| Changes from Initial Design | Changed most variables to utilise xna framework | |
| Variables | **NAME** | **USAGE** |
| Pos  Motion  Speed  Texture  scrBoundary | Holds the position  Holds the motion  Holds the speed  Holds the texture  Holds the boundary for the screen |
| Methods | **NAME** | **USAGE** |
| Bat  MovebatLeft  MoveBatRight  BoundaryCheck  GetBoundary  StartPosp1  StartPosP2  Draw | Constructor to create the bats  Moves the bat left  Moves the bat right  Checks to see if ithe player moves off scxreen  Draws a rectangle ontop of the bat  Sets the starting position of player 1  Sets the starting position of player 2  Draws the bat |

|  |  |  |
| --- | --- | --- |
| Class Name | BRICKS | |
| UsAGE | BRICKS | |
| Initial Design | BRICKS | |
| Changes from Initial Design | Changed most variable typing to utilise xna framework | |
| Variables | **NAME** | **USAGE** |
| Textture  Position  Colour  Visible  scorevalue | Holds the texture of the brick  Holds the position of the brick  Holds the colour of the brick  Holds whether or not the brick is visible  Holds the scorevalue for the brick |
| Methods | **NAME** | **USAGE** |
| Bricks  Position  CollisionCheck  Draw | Constructor for bricks  Retirns the positon of the ball  Checks if the brick has collided with the brick nad changes visibility  Draws the bricks |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Class Name | dbconn | |
| UsAGE | Loads and holds the values for the database | |
| Initial Design | New class | |
| Changes from Initial Design | n/a | |
| Variables | **NAME** | **USAGE** |
| Dra  Rowsrodraw  Dataset  Con  connectionString  trans  id | Holds the collection of rows  How many rows will be drawn  Holds the dataset  Holds the connection  Holds the position of the database in the file system  Holds the transaction  Holds the id |
| Methods | **NAME** | **USAGE** |
| DbConn  loadDb  checkRows  addtoDb  addNametodb | Constructor  Loads the database  Checks how many rows are in the database  Adds vaulse to the highscore table  Adds values to the player tables |

|  |  |  |
| --- | --- | --- |
| Class Name | GamePlayScreen | |
| UsAGE | Loads and runs the coop gamemode | |
| Initial Design | Game | |
| Changes from Initial Design | Added more methods and changed tpyings to utilise xna framework | |
| Variables | **NAME** | **USAGE** |
| Content  gameFont  P1Bat  P2Bat  ScreenBoundary  Ball  bricksWidth  brickImage  Bricks  dbConn  Score  Userinput  Gamemode  Lives  Random  Iswritten | holds the content pipeline  holds the font used  holds the bat used for p1  holds the bat used for p2  holds the screenboundary  holds the ball  holds how many bricks wide  holds howmany bricks high  holds the bricks  holds the database connection  holds the score  holds the user input  holds the gamemoed  holds the lives  holds random number generator  holds if the database had been written |
| Methods | **NAME** | **USAGE** |
| Gameplayscreen  loadContent  StartGame  UnloadContent  Update  HandleInput  Draw | Constructor  Loads the content that will be used within the game  Initilaises and creates objects required for game  Unloads the content  Updates the position and objects within the game  Handles the user input and control scheme  Draws the game |

|  |  |  |
| --- | --- | --- |
| Class Name | compGamePlay | |
| UsAGE | Loads and runs the competitive gamemode | |
| Initial Design | Game | |
| Changes from Initial Design | Added more methods and changed typings to utilise xna framework | |
| Variables | **NAME** | **USAGE** |
| Content  gameFont  P1Bat  P2Bat  ScreenBoundary  Ball  bricksWidth  brickImage  Bricks  dbConn  p1Score  Userinput  Gamemode  P1Lives  Random  Iswritten  Inputscore  P2score  Lastcollp1 | holds the content pipeline  holds the font used  holds the bat used for p1  holds the bat used for p2  holds the screenboundary  holds the ball  holds how many bricks wide  holds how many bricks high  holds the bricks  holds the database connection  holds the score  holds the user input  holds the gamemode  holds the lives  holds random number generator  holds if the database had been written  holds the score which will be inputted into the db  holds the score for player 2  holds if the last collision was with player 1  holds the lives for player 2 |
| Methods | **NAME** | **USAGE** |
| CoopGamePlay  loadContent  StartGame  UnloadContent  Update  HandleInput  Draw | Constructor  Loads the content that will be used within the game  Initilaises and creates objects required for game  Unloads the content  Updates the position and objects within the game  Handles the user input and control scheme  Draws the game |

## External Libraries

**Xna Framework is a .net framework created by Microsoft that was used as it is a very lightweight framework which is well documented and has plenty of resources available to use to aid in the creation of simple games in the c# language.**

|  |  |  |
| --- | --- | --- |
| Class Name | HighScoreScreen | |
| UsAGE | Loads and draws the database | |
| Initial Design | highscorescreen | |
| Changes from Initial Design | Added more methods and changed typings to utilise xna framework | |
| Variables | **NAME** | **USAGE** |
| dbConn  dbSelectMenuEntry  rowsrodraw  maxrows  dboptions  Options | Holds the database connection  Hols the menu entries  Holds how many rows to draw  Holds the max trows to draw  Holsd the options for which db to load  Holds the options for which db to load |
| Methods | **NAME** | **USAGE** |
| HighScoreScreen  SetMenuEntryText  DBSelect  Draw | Constructor  Sets the text of the menu option  Selects which database to load  Draws the highscores from the db |

**The classes in the following table where not created by me. They were obtained online and under the license MS-PL ( MicrosoftPublic license ) Copyright Grant- Subject to the terms of this license, including the license conditions and limitations in section 3, each contributor grants you a non-exclusive, worldwide, royalty-free copyright license to reproduce its contribution, prepare derivative works of its contribution, and distribute its contribution or any derivative works that you create. I am able to use these classes free of charge and royalty fee and make changes to them as I please**

|  |  |
| --- | --- |
| Classes | ScreenManage |
| UsAGE | the screen manager is a component which manages one or more GameScreen instances. It maintains a stack of screens, calls their Update and Draw methods at the appropriate times, and automatically routes input to thetopmost active screen. |
|  |  |
| Changes Made | None |
|  |  |

|  |  |
| --- | --- |
| Classes | LoadingScreen |
| UsAGE | The loading screen coordinates transitions between the menu system and the game itself. Normally one screen will transition off at the same time as the next screen is transitioning on, but for larger transitions that can take a longer time to load their data, we want the menu system to be entirely gone before we start loading the game. This is done as follows:  Tell all the existing screens to transition off.  Activate a loading screen, which will transition on at the same time.The loading screen watches the state of the previous screens.When it sees they have finished transitioning off, it activates the real next screen, which may take a long time to load its data. The loading screen will be the only thing displayed while this load is taking place. |
|  |  |
| Changes Made | None |
|  |  |

|  |  |
| --- | --- |
| Classes | MainMenuScreen |
| UsAGE | The main menu screen is the first thing displayed when the game starts up. |
|  |  |
| Changes Made | Changed the menu options so that they pointed to the respective screens properly and displayed the correct text also changed the background that is displayed |
|  |  |

|  |  |
| --- | --- |
| Classes | playerIndexEventARgs |
| UsAGE | Custom event argument which includes the index of the player who triggered the event. This is used by the MenuEntry.Selected event. |
|  |  |
| Changes Made | none |
|  |  |

|  |  |
| --- | --- |
| Classes | PauseMenuScreen |
| UsAGE | The pause menu comes up over the top of the game giving the player options to resume or quit. |
|  |  |
| Changes Made | none |
|  |  |

|  |  |
| --- | --- |
| Classes | MessageBoxScreen |
| UsAGE | A popup message box screen, used to display "are you sure? confirmation messages. |
|  |  |
| Changes Made | Changed the prompt on the box |
|  |  |

|  |  |
| --- | --- |
| Classes | MenuScreen |
| UsAGE | Base class for screens that contain a menu of options. The user can move up and down to select an entry, or cancel to back out of the screen. |
|  |  |
| Changes Made | none |
|  |  |

|  |  |
| --- | --- |
| Classes | MenuScreen |
| UsAGE | Base class for screens that contain a menu of options. The user can move up and down to select an entry, or cancel to back out of the screen. |
|  |  |
| Changes Made | none |
|  |  |

|  |  |
| --- | --- |
| Classes | MenuEntry |
| UsAGE | Helper class represents a single entry in a MenuScreen. By default this just draws the entry text string, but it can be customized to display menu entries in different ways. This also provides an event that will be raised when the menu entry is selected. |
|  |  |
| Changes Made | none |
|  |  |

|  |  |
| --- | --- |
| Classes | GameScreen |
| UsAGE | A screen is a single layer that has update and draw logic, and which can be combined with other layers to build up a complex menu system. For instance the main menu, the options menu, the "are you sure you want to quit" message box, and the main game itself are all implemented as screens. |
|  |  |
| Changes Made | none |
|  |  |

|  |  |
| --- | --- |
| Classes | Game |
| UsAGE | Loads the screenmanager and sets initial resolution |
|  |  |
| Changes Made | none |
|  |  |

# Database Design

ERD

|  |  |
| --- | --- |
|  | # - indicates the primary key  \* - indicates a not-null value  Indicates a mandatory one to many (1:M) relationship |
| Player MUST have one or more than one Score  Score MUST have one and only one Player | |

Player

User

#Id

\*name

HighScores

#id

‘\*score

\*mode

|  |  |  |  |
| --- | --- | --- | --- |
| Database name | type | optionality | Description |
| Id | Integer (PK) | Not Null | Holds the id for the player |
| name | String | Not Null | Holds the name of the player |

## High Scores

|  |  |  |  |
| --- | --- | --- | --- |
| Database name | type | optionality | Description |
| id | Integer (FK/PK) | Not Null | Id for the highscore |
| score | Integer | Not Null | Value for the score |
| user\_id | Integer{(fk) | Notnull | Pulls the id from player |
| mode | Strine | Notnull | The mode of the game |

## SQL Commands"SELECT Player.Name, HighScores.Score, HighScores.ModeType FROM (HighScores INNER JOIN Player ON HighScores.[user\_id] = Player.ID AND HighScores.[user\_id] = Player.ID) WHERE (HighScores.ModeType = 'CoOp') ORDER BY HighScores.Score DESC, Player.Name;"

## loads the coop mode and performs a join on the playerid = highscore id

"SELECT Player.Name, HighScores.Score, HighScores.ModeType FROM (HighScores INNER JOIN Player ON HighScores.[user\_id] = Player.ID AND HighScores.[user\_id] = Player.ID) WHERE (HighScores.ModeType = 'Comp') ORDER BY HighScores.Score DESC, Player.Name;"

loads the comp mode and performs a join on the player id = highscore id

"INSERT INTO Player(Name) VALUES(@Name);"

inserts the value @name into the playername table

"SELECT @@IDENTITY;"

returns the value autogenerated value for the last thing inserted into the databse

"INSERT INTO HighScores(Score,ModeType,user\_id) VALUES(@Score,@Mode,@id);";

inserts into the highscore table

## Testing

## Methodology

**I chose to use the whitebox method of testing due to time constraints if given more time I would have liked to create unit tests for each method.**

## Testing plan

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case | Expected Results | Actual Results | Comments |
|  |  |  |  |
|  |  |  |  |

Due to time constraints and physical limitations, thorough testing of the game was not possible, and was mostly limited to the PC it was coded on.

Ideally, testing would be performed on numerous machines of varying system specifications, and test runs would have a wider range of pre-determined cases.

## Test case

|  |  |  |  |
| --- | --- | --- | --- |
| TestCase | Expected | Actual | Comments |
| Click start game | Game starts | Game starts | Works As intended |
| Chose highscore menu | Highscore menu loads and database draws | Highscore menu loads and database draws | Works As intended |
| Chose competitive mode | Competitive mode starts | Competitive mode starts | Works As intended |
| Chose exit | Game exits | Game exits | Works As intended |
| Press key to move player 1 left | Player 1 moves left | Player 1 moves left | Works As intended |
| Press key to move player 1 right | Player 1 moves right | Player 1 moves right | Works As intended |
| Press key to move player 2 right | Player 2 moves right | Player 2 moves right | Works As intended |
| Press key to move player 2 left | Player 2 moves left | Player 2 moves left | Works As intended |
| Press player 1 key to start | Ball starts moving | Ball starts moving | Works As intended |
| Press key to pause game | Pause menu shows up | Pause menu shows up | Works As intended |
| Ball hits bat | Ball should change velocity and deflect off | Ball changes velocity and deflects off | Works As intended |
| Ball hits bricks | Ball should change velocity and deflect off and add points to score brick should disappear | Ball changes velocity and deflecs off and add points to score brick dissapears | Works As intended |
| Ball falls off screen | Resets the ball position and deducts life | Reset ball position and deducts life | Works As intended |
| Ball hits edge of screen | Ball changes velocity and deflects off | Ball changes velocity and changes off | Works As intended |
| Bat gets to edge of screen | Ball shouldnt be able to leave the screen | Bat cant leave the screen | Works As intended |
| Upon lives hitting 0 | Game should stop and ask user to input name for score | Game stops displays gameover and asks user to input score | Works As intended |
| Write score to database | Score should write to database with the name of the player | Score and name write to the database | Works As intended |
| Load database | Loads the database and maps it to variablse | Loads to the database and maps it to variableds | Works As intended |
| Draws database on highscore screen | Draws database on highscore screen with mode that is selected | Draws database on highscore screen with mode that is selected | Works As intended |
| Change the mode | Should remove what is drawn and redraw the new entires | Removes what is already drawn and redraws the new entries | Works As intended |

# Appendix

## Data Dictionary

**FILL THIS IN.**

|  |  |  |  |
| --- | --- | --- | --- |
| Attribute | Data Type | Location | Description |
| DRA | DATAROWCOLLECTION | DBCONN | MAPS THE DATA TO A DATA ROW COLLECTON |
| ROWSRODRAW | INT | DBCONN | HOW MANY ROOWS TO DRAW |
| DATASET | DATASET | DBCONN | CREATES A NEW DATASET |
| CON | OLEDBCONNECTION | DBCONN | CREATES A NEW CONNECTION VALUE |
| CONNECTIONSTRING | STRING | DBCONN | THE POSITION AS TO WHERE THE DATABASE IS ON THE FILESYSTEM |
| TRANS | OLEDBTRANSACTION | DBCONN | HOLDS THE VLAUE FOR THE TRANSACTION |
| ID | INT | DBCONN | HOLDS THE INTEGER ID |
| CONTENT | CONTENTMANAGER | COMPGAMEPLAY | HOLDS THE CONTENTMANAGER FOR THE GAME |
| GAMEFONT | SPRITEFONT | COMPGAMEPLAY | HOLDS THE FONT FOR THE GAME |
| P1BAT | BAT | COMPGAMEPLAY | HOLDS THE BAT |
| P2BAT | BAT | COMPGAMEPLAY | HOLDS THE PLAYER 2 BAT |
| SCREENBOUNDARY | RECTANGLE | COMPGAMEPLAY | HOLDS THE SCREENBOUNDARIES |
| BALL | BALL | COMPGAMEPLAY | HOLDS THE BALL |
| BRICKSWIDTH | INT | COMPGAMEPLAY | HOLDS HOW MANY BRICKS WIDE |
| BRICKSHEIGHT | INT | COMPGAMEPLAY | HOLDS HOW MANY BRICKS HIGH |
| BRICKIMAGE | TEXTURE2D | COMPGAMEPLAY | HOLDS THE TEXTURE FOR THE BRICKS |
| BRICKS | BRICKS | COMPGAMEPLAY | HOLDS THE BRICKS |
| INPUTSCORE | INT | COMPGAMEPLAY | HOLDS THE SCORE TO INPUT IN TO THE DATABASE |
| DBCONN | DBCONN | COMPGAMEPLAY | HOLDS THE DATABASE CONNECTONI |
| P1SCORE | INT | COMPGAMEPLAY | HOLDS THE SCORE FOR PLAYER1 |
| P2SCORE | INT | COMPGAMEPLAY | HOLDS THE PLAYER 2 |
| USERINPUT | STRING | COMPGAMEPLAY | HOLDS THE STRING WHICH WILL BE INPUTED TO THE DATABASE |
| GAMEMODE | STRING | COMPGAMEPLAY | HOLDS WHICH DATABASE TO LOAD |
| P1LIVES | INT | COMGAMEPLAY | HOLDS THE LIVES FOR PLAYER ONE |
| P2LIVES | INT | COMPGAMEPLAY | HOLDS THE LIVES FOR PLAYER 2 |
| LASTCOLLP1 | BOOL | COMPGAMEPLAY | HOLDS IF THE LAST COLLISION WAS PLAYER 1 |
| RANDOM | RANDOM | COMPGAMEPLAY | HOLDS THE RANDOM NUMBER GENERATOR |
| ISWRITEN | BOOL | COMPGAMEPLAY | HOLDS IF THE DATABASE HAS BEEN WRITTN |
| CONTENT | CONTENTMANAGER | COOPGAMEPLAY | HOLDS THE CONTENTMANAGER FOR THE GAME |
| GAMEFONT | SPRITEFONT | COOPGAMEPLAY | HOLDS THE FONT FOR THE GAME |
| P1BAT | BAT | COOPGAMEPLAY | HOLDS THE BAT |
| P2BAT | BAT | COOPGAMEPLAY | HOLDS THE PLAYER 2 BAT |
| SCREENBOUNDARY | RECTANGLE | COOPGAMEPLAY | HOLDS THE SCREENBOUNDARIES |
| BALL | BALL | COOPGAMEPLAY | HOLDS THE BALL |
| BRICKSWIDTH | INT | COOPGAMEPLAY | HOLDS HOW MANY BRICKS WIDE |
| BRICKSHEIGHT | INT | COOPGAMEPLAY | HOLDS HOW MANY BRICKS HIGH |
| BRICKIMAGE | TEXTURE2D | COOPGAMEPLAY | HOLDS THE TEXTURE FOR THE BRICKS |
| BRICKS | BRICKS | COOPGAMEPLAY | HOLDS THE BRICKS |
| INPUTSCORE | INT | COOPGAMEPLAY | HOLDS THE SCORE TO INPUT IN TO THE DATABASE |
| DBCONN | DBCONN | COOPGAMEPLAY | HOLDS THE DATABASE CONNECTONI |
| SCORE | INT | COOPGAMEPLAY | HOLDS THE SCORE FOR PLAYER1 |
| USERINPUT | STRING | COOPGAMEPLAY | HOLDS THE STRING WHICH WILL BE INPUTED TO THE DATABASE |
| GAMEMODE | STRING | COOPGAMEPLAY | HOLDS WHICH DATABASE TO LOAD |
| P1LIVES | INT | COOPGAMEPLAY | HOLDS THE LIVES FOR PLAYER ONE |
| LASTCOLLP1 | BOOL | COOPGAMEPLAY | HOLDS IF THE LAST COLLISION WAS PLAYER 1 |
| RANDOM | RANDOM | COOPGAMEPLAY | HOLDS THE RANDOM NUMBER GENERATOR |
| ISWRITEN | BOOL | COOPGAMEPLAT | HOLDS IF THE DATABASE HAS BEEN WRITTN |
| DBCONN | DBCONN | HIGHSCORESCREEN | HOLDS THE DATABASE CONNECTION |
| DBSELECTMENUENTRY | MENUENTRY | HIGHSCORESCREEN | HOLDS THE MENU ENTRY |
| ROWSRODRAW | INT | HIGHSCORESCREEN | HOW MANY ROWS TO DRAW |
| MAXROWS | INT | HIGHSCORESCREEN | MAX AMMOUNT OF ROWS TO DRAW |
| DBOPTIONS | STRING[] | HIGHSCORESCREEN | THE OPTIONS OF THE HIGHSCORE DB TO LAOD |
| MOTION | VECTOR2 | BALL | HOLDS THE MOTION OF THE BALL |
| POS | VECTOR2 | BALL | HOLDS THE MOTION OF THE BALL |
| SPEED | VECTOR2 | BALL | HOLDS THE POSITION OF THE BALL |
| TEXTURE | TEXTURE2D | BALL | HOLDS THE TEXTURE OF THE BALL |
| SCRBOUNDARY | RECTANGLE | BALL | HOLDS THE BOUNDARY OF THE SCREEN |
| BOUNDARY | RECTANGLE | BALL | HOLDS THE BOUNDARY OF THE BALL |
| INITIALSPEED | FLOAT | BALL | HOLDS THE SPEED OF THE BALL |
| RANDOM | RANDOM | BALL | HOLDS THE RANDOM NUMBER GENERATOR |
| COLLISION | BOOL | BALL | HOLDS IF THE BALL HAS COLLIDIED |
| POS | VECTOR2 | BAT | HOLDS THE POSITION OF THE BAT |
| MOTION | VECTOR2 | BAT | HOLDS THE MOTION FO THE BAT |
| SPEED | VECTOR2 | BAT | HOLDS THE SPEED OF THE BAT |
| TEXTURE | TEXTURE2D | BAT | HOLDS THE TEXTURE OF THE BAT |
| SCRBOUNDARY | TEXTURE2D | BAT | HOLDS THE BOUNDARY FOR THE SCREEN |
| TEXTURE | TEXTURE2D | BRICKS | HOLDS THE TEXTURE FOR THE BRICKS |
| POSITION | RECTANGLE | BRICKS | HOLDS THE POSITION OG THE BRICK |
| COLOUR | COLOR | BRICKS | HOLDS THE COLOUR OF THE BRICKS |
| VISIBLE | BOOL | BRICKS | HOLDS IF THE BRICKS ARE VISIBLE |
| SCOREVALUE | INT | BRICKS | HOLDS THE SCOREVALUE FOR THE BRICKS |
|  |  |  |  |

## Code

Sent as a.zip file