World Defence:

Technical Manual

Version 1.0

18/04/2015

Created By: Siobhan Wallace

Table of Contents

[Brief 3](#_Toc418033936)

[System Requirements 3](#_Toc418033937)

[Hardware Requirements 3](#_Toc418033938)

[Software Requirements 3](#_Toc418033939)

[Functionality 4](#_Toc418033940)

[Implementation Comments 4](#_Toc418033941)

[Additional Functionality 5](#_Toc418033942)

[Design 6](#_Toc418033943)

[Classes 6](#_Toc418033944)

[Class Diagram 17](#_Toc418033945)

[External Libraries 17](#_Toc418033946)

[High-Score Database 18](#_Toc418033947)

[ERD 18](#_Toc418033948)

[Database Design 18](#_Toc418033949)

[Score 18](#_Toc418033950)

[Player 18](#_Toc418033951)

[SQL Commands 19](#_Toc418033952)

[Testing 19](#_Toc418033953)

[Testing Methodology 19](#_Toc418033954)

[Test Plan 19](#_Toc418033955)

[Testing Cases 20](#_Toc418033956)

[Test Player Feedback 22](#_Toc418033957)

[Appendix 23](#_Toc418033958)

[Data Dictionary 23](#_Toc418033959)

[Full Code 23](#_Toc418033960)

## Brief

As part of my HND Software Development: Year Two Graded Unit, I propose to create a 2D arcade game. This game will be based off the 1978 classic ‘Space Invaders’, and in which the user may defeat waves of enemies by controlling a turret.

This game will be written in the C# programming language, using MS Visual Studio as the development environment, with the addition of Microsoft’s XNA framework.

As the proposed game will be of the Fixed Shooter genre, it will be following these basic rules;

1. The player must defeat waves of mobs in order to advance to the next level
2. The player is defeated if they either fail to stop the mobs or are killed.

## System Requirements

The game UI will be designed around keyboard input, and all graphical elements should be reasonably comfortable to use.

The game has a simple design and as such should run on any modern pc. A keyboard and mouse is required in order to fully interact with the game.

### Hardware Requirements

|  |  |  |
| --- | --- | --- |
|  | Minimum | Recommended |
| **Operating System:** | Windows XP with SP2 | Windows 7 or 8 |
| **Processor:** | 1GHz |  |
| **Graphics Card:** | Support for Shader Model 1.1 | Card with 512mb or better |
| **Memory:** | 512MB | 1GB or better |
| **Sound Card:** | N/A | Standard card or better |
| **Input:** | Keyboard | Keyboard / Xbox Controller |

Software Requirements

As World Defence uses Microsoft’s XNA framework, it requires either XNA or its redistributable to be installed on the client machine. The installer for World Defence automatically installs this for the user.  
In order for the high score database to function, either Microsoft Access or Microsoft Access Runtime must be installed.

Microsoft Access Runtime can be found here: <http://www.microsoft.com/en-gb/download/details.aspx?id=39358>

### Functionality

|  |  |  |
| --- | --- | --- |
| Functional Requirements | | |
| Number | **Description** | **Implemented** |
| 1 | The Player will be shown game menu on application start | **Yes** |
| 2 | The Player may exit the application at any time | **Yes** |
| 3 | The Player may quit the game at any time | **Yes** |
| 4 | The Player may pause the game at any time | **Yes** |
| 5 | The Player may exit to the main menu at any time | **Yes** |
| 6 | When the Player chooses to exit, a confirmation window will appear | **Yes** |
| 7 | The Player will “die” if hit 3 times during the game |  |
| 8 | The Player must defeat 11 ‘waves’ of enemies |  |
| 9 | The Boss will spawn after the 10th ‘wave’ of enemies |  |
| 10 | The Player will be able to toggle game sound on/off | **Yes** |
| 11 | Music will be playing when Game is running | **Yes** |
| 12 | Visual and Auditory effects will play when the Player is hit. |  |
| 13 | The Player will be prompted to enter their name on Game Over |  |

|  |  |  |
| --- | --- | --- |
| Non-Functional Requirements | | |
| Number | **Description** | **Implemented** |
| 14 | The Game will run on the Windows platform, with support primarily being aimed at Windows 7 and 8. | **Yes** |
| 15 | Game background will be black. | **No** |
| 16 | The High Score will only display top 10 entries |  |
| 17 | The Player will be awarded points for each normal enemy eliminated | **Yes** |
| 18 | The Player will be awarded 100 points for eliminating the Boss |  |
| 19 | Each enemy will come in one of the following colours (red, purple, yellow, orange, green) | **Partially** |

### Implementation Comments

|  |  |
| --- | --- |
| Number | Reason |
|  |  |
| 15 | A minimalistic background was implemented instead of a plain black background, as it was decided that it makes the game look better. |
| 19 | Enemies come in 5 different types, and each type is a different colour, which aids the player in identification. It was decided that additional colour differences would only confuse the player |

### Additional Functionality

I totally added extra stuff. Because I got my initial shit totally done. Yeah.

## Design

### Classes

|  |  |  |
| --- | --- | --- |
| Class Name | Player | |
| Use | Holds and initialises all variables for the player | |
| Initial Design |  | |
| Changes from Initial Design |  | |
| Variables | **Name** | **Use** |
| playerCoords | Holds the current position of the player |
| movement | Holds the current movement of the player |
| ccreenBoundary | Holds boundary of the screen to detect collision |
| isAlive | Stores if player is alive; to determine game over |
| playerShield | Holds the players current shield level, between 3 and 0 |
| playerScore | Holds the players current score |
| playerSprite | Texture for the player |
| playerSpeed | Holds the player speed |
| Methods | **Name** | **Use** |
| GetPCoords() | Gets the players current position on screen |
| AddScore() | Sets the players current score |
| GetBoundary() | Gets the players boundary to detect collision |
| Player() | Constructor for the player; sets initial values |
| MovePlayerLeft() | Handles moving the player left |
| MovePlayerRight() | Handles moving the player right |
| ScrBoundaryCheck() | Checks if the player has left the screen boundaries |
| Draw() | Draws the player on screen |

|  |  |  |
| --- | --- | --- |
| Class Name | Lasers | |
| Use |  | |
| Initial Design |  | |
| Changes from Initial Design |  | |
| Variables | **Name** | **Use** |
| boundary |  |
| isActive |  |
| laserPos |  |
| laserSpeed |  |
| playerLaserTex |  |
| movement |  |
| screenBoundary |  |
| Methods | **Name** | **Use** |
| Draw() |  |
| Fire() |  |
| HasExited() |  |
| IfIsActive() |  |
| Lasers() |  |
| SetIsActive() |  |
| SetLaserColour() |  |
| Update() |  |
| UpdatePosition() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | Enemies | |
| Use |  | |
| Initial Design |  | |
| Changes from Initial Design |  | |
| Variables | **Name** | **Use** |
| enemyPos |  |
| enemySprite |  |
| isAlive |  |
| scoreValue |  |
| Methods | **Name** | **Use** |
| CollisionCheck() |  |
| Draw() |  |
| Enemies() |  |
| GetIsAlive() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | PlayerIndexEventArgs | |
| Use |  | |
| Initial Design | Was not in the original design | |
| Changes from Initial Design | N/A | |
| Variables | **Name** | **Use** |
| playerIndex |  |
| Methods | **Name** | **Use** |
| PlayerIndexEventArgs() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | ArcadeGameplay | |
| Use |  | |
| Initial Design |  | |
| Changes from Initial Design |  | |
| Variables | **Name** | **Use** |
| bgTex |  |
| content |  |
| enemies |  |
| enemy1Sprite |  |
| enemy2Sprite |  |
| enemy3Sprite |  |
| enemy4Sprite |  |
| enemyHeight |  |
| enemyLasers |  |
| enemyLaserSound |  |
| enemyLaserTex |  |
| enemyWidth |  |
| fireDelay |  |
| gamefont |  |
| gameMusic |  |
| gameMusicInstance |  |
| playerLaserTex |  |
| pauseAlpha |  |
| player |  |
| playerLasers |  |
| playerlaserSound |  |
| prevFiretime |  |
| scoreValue |  |
| screenBoundary |  |
| Methods | **Name** | **Use** |
| AddEnemy() |  |
| AddLaser |  |
| ArcadeGamePlay() |  |
| Draw() |  |
| HandleInput() |  |
| Initialize() |  |
| LoadContent() |  |
| PlaySound() |  |
| UnloadContent() |  |
| Update() |  |
| UpdateCollision() |  |
| UpdateEnemies() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | ArcadeGameplay | |
| Use |  | |
| Initial Design |  | |
| Changes from Initial Design |  | |
| Variables | **Name** | **Use** |
| bgTex |  |
| content |  |
| enemies |  |
| enemy1Sprite |  |
| enemy2Sprite |  |
| enemy3Sprite |  |
| enemy4Sprite |  |
| enemyHeight |  |
| enemyLasers |  |
| enemyLaserSound |  |
| enemyLaserTex |  |
| enemyWidth |  |
| fireDelay |  |
| gamefont |  |
| gameMusic |  |
| gameMusicInstance |  |
| playerLaserTex |  |
| pauseAlpha |  |
| player |  |
| playerLasers |  |
| playerlaserSound |  |
| prevFiretime |  |
| scoreValue |  |
| screenBoundary |  |
| Methods | **Name** | **Use** |
| AddEnemy() |  |
| AddLaser |  |
| ArcadeGamePlay() |  |
| Draw() |  |
| HandleInput() |  |
| Initialize() |  |
| LoadContent() |  |
| PlaySound() |  |
| UnloadContent() |  |
| Update() |  |
| UpdateCollision() |  |
| UpdateEnemies() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | CoopGameplay | |
| Use |  | |
| Initial Design |  | |
| Changes from Initial Design |  | |
| Variables | **Name** | **Use** |
| bgTex |  |
| bossCurrentHP |  |
| bossLaserTex |  |
| bossPos |  |
| bossSprite |  |
| content |  |
| enemies |  |
| enemy1Sprite |  |
| enemy2Sprite |  |
| enemy3Sprite |  |
| enemy4Sprite |  |
| enemyHeight |  |
| enemyLasers |  |
| enemyLaserSound |  |
| enemyLaserTex |  |
| enemyWidth |  |
| explosions |  |
| fireDelay |  |
| gameFont |  |
| gameMusic |  |
| gameMusicInstance |  |
| playerLaserTex |  |
| pauseAlpha |  |
| player |  |
| player2 |  |
| player2Pos |  |
| player2Sprite |  |
| playerLasers |  |
| playerLaserSound |  |
| prevFireTime |  |
| scoreValue |  |
| screenBoundary |  |
| Methods | **Name** | **Use** |
| AddEnemy() |  |
| AddLaser() |  |
| Draw |  |
| CoopGameplay() |  |
| HandleInput() |  |
| Initialize() |  |
| LoadContent() |  |
| PlaySound() |  |
| UnloadContent() |  |
| Update() |  |
| UpdateBoss() |  |
| UpdateEnemies() |  |
| UpdateCollision() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | GameScreen | |
| Use | Holds all the generic attributes for loading the game screens and acts as the parent class for each screen | |
| Initial Design | Was not in the original design | |
| Changes from Initial Design | N/A | |
| Variables | **Name** | **Use** |
| conPlayer |  |
| isExiting |  |
| isPopup |  |
| otherScreenHasfocus |  |
| screenManager |  |
| screenState |  |
| transOffTime |  |
| transOnTime |  |
| transPos |  |
| Methods | **Name** | **Use** |
| ConPlayer() |  |
| IsActive() |  |
| IsExiting() |  |
| IsPopup() |  |
| ScreenManager() |  |
| ScreenState() |  |
| TransAlpha() |  |
| TransOffTime() |  |
| TransOnTime() |  |
| TransPos() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | InputState | |
| Use |  | |
| Initial Design | Was not in the original design | |
| Changes from Initial Design | N/A | |
| Variables | **Name** | **Use** |
| currentGamePadStates |  |
| currentKeyboardSTates |  |
| gamePadWasConnected |  |
| lastGamePadStates |  |
| lastKeyboardStates |  |
| maxInputs |  |
| Methods | **Name** | **Use** |
| InputState() |  |
| IsFired() |  |
| IsMenuCancel() |  |
| IsMenuDown() |  |
| IsMenuUp() |  |
| IsMenuSelect() |  |
| IsNewButtonPress() |  |
| IsNewKeyPress() |  |
| IsPauseGame() |  |
| Update() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | ScreenManager | |
| Use |  | |
| Initial Design | Was not in the original design | |
| Changes from Initial Design | N/A | |
| Variables | **Name** | **Use** |
| blankTexture |  |
| font |  |
| input |  |
| isInitialized |  |
| sBatch |  |
| screens |  |
| screensToUpdate |  |
| Methods | **Name** | **Use** |
| AddScreen() |  |
| Draw() |  |
| FadeBackBufferToBlack() |  |
| GetScreens(Initialize()) |  |
| LoadContent() |  |
| RemoveScreen() |  |
| ScreenManager() |  |
| UnloadContent() |  |
| Update() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | HighscoreScreen | |
| Use |  | |
| Initial Design |  | |
| Changes from Initial Design |  | |
| Variables | **Name** | **Use** |
| currentMode |  |
| dataInt |  |
| gameMode |  |
| maxRows |  |
| modeEntry |  |
| rowDraw |  |
| Methods | **Name** | **Use** |
| HighscoreScreen() |  |
| LoadContent() |  |
| ModeEntrySel() |  |
| SetMenuText() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | Loading | |
| Use | Holds all the variables required for the loading screen and acts as the loading screen | |
| Initial Design | Was not in the original design | |
| Changes from Initial Design | N/A | |
| Variables | **Name** | **Use** |
| loadingIsSlow | Holds if loading is taking a while, which determines if the loading message should be shown |
| otherScreensGone | Holds if the other screens have transitioned off |
| screensToLoad | Holds which screens are to be loaded |
| Methods | **Name** | **Use** |
| Draw() | Draws the loading screen |
| Load() | Activates the loading screen |
| Loading() | Constructor for the loading screen |
| Update() | Updates the loading screen |

|  |  |  |
| --- | --- | --- |
| Class Name | MainMenu | |
| Use | Holds all the variables required and acts as the main menu screen | |
| Initial Design |  | |
| Changes from Initial Design |  | |
| Variables | **Name** | **Use** |
| None | N/A |
| Methods | **Name** | **Use** |
| MainMenu() | Constructor for the main menu |
| EndlessModeMenuEntrySel() | Event handler for when endless Mode entry is selected |
| ArcadeModeMenuEntrySel() | Event handler for when arcade Mode entry is selected |
| CoopModeMenuEntrySel() | Event handler for when co-op Mode entry is selected |
| OptionsMenuEntrySel() | Event handler for when options entry is selected |
| HighscoreMenuEntrySel() | Event handler for when high score entry is selected |
| OnCancel() | When user wants to exit, asks for confirmation |
| ConfirmExitMessageBoxAccepted() | Event handler for when user selects ok on message box |

|  |  |  |
| --- | --- | --- |
| Class Name | MenuEntry | |
| Use | Holds the attributes needed for the menu entries and acts as the parent class to create new menu entries | |
| Initial Design | Was not in the original design | |
| Changes from Initial Design | N/A | |
| Variables | **Name** | **Use** |
| pos |  |
| selFade |  |
| text |  |
| Methods | **Name** | **Use** |
| Draw() |  |
| GetHeight() |  |
| GetWidth() |  |
| MenuEntry() |  |
| OnSelectEntry() |  |
| Update() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | MenuScreen | |
| Use | Holds all the generic variables required to load each menu screen and acts as the parent class for each menu | |
| Initial Design | Was not in the original design | |
| Changes from Initial Design | N/A | |
| Variables | **Name** | **Use** |
| content | Loads a new content manager for the menu content |
| menuEntries | Creates menu entries as a new menu entry list |
| menuMusic | Holds the menu music |
| menuMusicInstance | Holds the menu music |
| menuTitle | Holds the menu title |
| selEntry | Holds the integer for entry selection |
| Methods | **Name** | **Use** |
| Draw() | Draws the menu screens |
| HandleInput() | Handles player input for the menu screens |
| LoadContent() | Loads content for the menu screens |
| MenuScreen() | Constructor for the menu screen |
| Update() | Updates the menu screens |
| UpdateMenuEntries() | Updates the menu entries |
| OnSelectEntry() | Event handler for when the player selects a menu entry |
| OnCancel() | Event handler for when the player selects cancel |

|  |  |  |
| --- | --- | --- |
| Class Name | MsgboxScreen | |
| Use | Holds all the variables associated with the message box and acts as the message box screen | |
| Initial Design | Was not in the original design | |
| Changes from Initial Design | N/A | |
| Variables | **Name** | **Use** |
| message | Holds the message to be displayed |
| gradientTexture | Holds the texture for the background of the message box |
| Methods | **Name** | **Use** |
| Draw() | Draws the message box on screen |
| HandleInput() | Accepts user input for the message box |
| LoadContent() | Loads the screen content |
| MsgboxScreen() | The constructor for the message box |

|  |  |  |
| --- | --- | --- |
| Class Name | OptionsScreen | |
| Use | Holds all the variables associated with the options menu | |
| Initial Design |  | |
| Changes from Initial Design |  | |
| Variables | **Name** | **Use** |
| currentColourloading | Holds the currently shown colour option |
| currentMusic | Holds the currently shown music option |
| currentShip | Holds the currently shown ship option |
| laserColourEntry | Initialises laserColour as a MenuEntry class |
| musicEntry | Initialises music as a MenuEntry class |
| playerShipEntry | Initialises playership as a MenuEntry class |
| Methods | **Name** | **Use** |
| LaserColourEntrySel() | Toggles through the laser colour options when selected |
| MusicEntrySel() | toggles the music on or off when selecte |
| PlayerShipEntrySel() | Toggles through the player ship options when selected |
| LoadContent() | Loads the content for this menu |
| OptionsScreen() | Constructor for the options screen |
| SetMenuText() | Sets the menu texts |

|  |  |  |
| --- | --- | --- |
| Class Name | PauseScreen | |
| Use | Holds all the variables associated with the pause screen and acts as the games pause screen | |
| Initial Design | Was not in the original design | |
| Changes from Initial Design | N/A | |
| Variables | **Name** | **Use** |
| None | N/A |
| Methods | **Name** | **Use** |
| PauseScreen() | Constructor for the pause screen |
| ConfirmMsgBoxAccepted() | Handles the user selecting quit game event |
| QuitGameMenuEntrySel() | Handles user confirming quit event |

|  |  |  |
| --- | --- | --- |
| Class Name | DatabaseInt | |
| Use | Holds all the variables required for interacting with the database and acts as the games link to the database | |
| Initial Design |  | |
| Changes from Initial Design |  | |
| Variables | **Name** | **Use** |
| connection |  |
| conString |  |
| dataRowC |  |
| dataset |  |
| playerID |  |
| rowDraw |  |
| transaction |  |
| Methods | **Name** | **Use** |
| AddNameDB() |  |
| CheckRows() |  |
| DataBaseInt() |  |
| Draw() |  |
| ReadDatabase() |  |
| WriteDatabase() |  |

|  |  |  |
| --- | --- | --- |
| Class Name | Game | |
| Use | The main class of the game | |
| Initial Design |  | |
| Changes from Initial Design |  | |
| Variables | **Name** | **Use** |
| Graphics |  |
| preloadAssets |  |
| sBatch |  |
| screenManager |  |
| Methods | **Name** | **Use** |
| Draw() |  |
| Game() |  |
| Initialise() |  |
| LoadContent() |  |
| UnloadContent() |  |
| Update() |  |

### Class Diagram

[insert]

### External Libraries

Microsoft’s s XNA

## High-Score Database

### ERD

|  |  |
| --- | --- |
|  | # - indicates primary ley  \* - 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 | |

### Database Design

### Score

Primary Key: Rank

Foreign Key: player\_ID

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Database name | Optionality | Data type | Description |
|  | Rank | Not Null | Int(2) | Stores the rank achieved by the player  (between 1-10) |
|  | Player\_ID | Not Null | Int(4) | Stores the unique number assigned to each player |
|  | Score | Not Null | Int(16) | Stores the score achieved by the player |
|  | Mode | Not Null | String | Stores which game mode was played |

### Player

Primary Key: player\_ID

Foreign Key: N/A

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Database name | Optionality | Data type | Description |
|  | Player\_ID | Not Null | Int(4) | Stores the unique number assigned to each player |
|  | Name | Not Null | String | Stores the player name |

### SQL Commands

|  |  |
| --- | --- |
|  |  |
| SQL Command | **Use** |
|  |  |
|  |  |
|  |  |
|  |  |

## Testing

### Testing Methodology

Testing for World Defence was split between white-box and black-box testing. Due to constraints on time and limited resources available, the black-box testing was done on very few machines. The machine it was coded on, a desktop which runs Windows 7 Ultimate, 64bit, was used for this purpose.

In an ideal situation, testing would be conducted more thoroughly, using many other machines to fully test compatibility.

Test cases have been designed to test functionality, with the expected and actual results being documented. Any bugs found will be dealt with before another test session takes place to note if the problems have been resolved.

### Test Plan

Testing will be performed on the major areas of the game: The Menu System, the High score System and the three Game Modes. Due to time constraints, Integration testing will be done for the UI, taking in the menu system as a whole, rather than individual unit tests per screen. However, Unit testing will be conducted for each Game Mode, and the Database Integration, with each Unit being taken as the class, due to the game being object oriented.

An example testing case form is shown below;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| System Testing | | | | |
| Test | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
|  |  |  |  |  |
|  |  |  |  |  |

### Testing Cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Menu System Testing | | | | |
| Test – main screen | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
| Can the user access the high score menu? | Player can access high score menu |  |  |  |
| Can the user access the options menu? | Player can access options menu |  |  |  |
| Can the user access the endless game mode? | Player can access endless mode |  |  |  |
| Can the user access the co-op game mode? | Player can access co-op mode |  |  |  |
| Can the user access the arcade game mode? | Player can access arcade mode |  |  |  |
| Test – Pause Menu | | | | |
| Can the user pause the game at any time? | Player can pause at any time |  |  |  |
| Can the user quit the game from the pause menu? | Player can quit from pause screen |  |  |  |
| Can the user return to the main menu? | Player can return to main menu |  |  |  |
| Test – Options Menu | | | | |
| Can the user change their laser colour? | Player can change laser colour |  |  |  |
| Can the user change their model? | Player can change model |  |  |  |
| Can the user toggle sound on/off? | Player can toggle sound |  |  |  |
| Test – All Menu’s | | | | |
| Can the user return to the main menu from any of the above screens? | Player can return to the main menu |  |  |  |
| Does the menu music correctly play in each of the above menus? | Menu music plays correctly |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| High score System Testing | | | | |
| Test | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
| Can player enter their name? | Player can enter their name |  |  |  |
| Can the player choose which game scores to display? | Player can choose which game mode they would like to have scores displayed for |  |  |  |
| Does the game correctly retrieve the scores? | Game correctly shows score table |  |  |  |
| Does the database store more than 10 entries? | Database does not store more than 10 entries |  |  |  |
| Does the game correctly show which rank was achieved? | Game correctly shows achieved rank |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Endless Mode Testing | | | | |
| Test | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
| Can the player exit the game-screen boundary? | Player cannot exit the game-screen boundary. |  |  |  |
| Does the game end if the player’s health drops to 0? | Game ends if player health drops to 0 |  |  |  |
| Does the game end if the player fails to stop the enemies? | Game ends if player fails to stop enemies |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Co-Op Mode Testing | | | | |
| Test | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
| Can the players exit the game-screen boundary? | Players cannot exit the game-screen boundary. |  |  |  |
| Does each player have their own health? | Each player has their own health |  |  |  |
| Does each player have their own individual score? | Each player has their own individual score |  |  |  |
| Can each player move independently of the other? | Players can move independently of each other |  |  |  |
| Does the game end if any player dies? | Game ends if any player dies |  |  |  |
| Does the game end if the players fail to stop the enemies? | Game ends if players fail to stop enemies |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Arcade Mode Testing | | | | |
| Test | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments** |
| Can the player exit the game-screen boundary? | Player cannot exit the game-screen boundary. |  |  |  |
| Does the game end if the player’s health drops to 0? | Game ends if player health drops to 0 |  |  |  |
| Does the game end if the player fails to stop the enemies? | Game ends if player fails to stop enemies |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Test Player Feedback

## Appendix

### Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Data Type | Accessibility | Location | Description |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

### Full Code