

Software Design Specification

for

Universe of Things (*UOT*)

Tempus Fugit
2/27/17

Prepared by: John G. Toland
Richard O'Neal
Dylan Salopek
Nicholas Muirhead
Gerard Fierro

Table of Contents

Project Description	3
System Analysis and Decomposition	4
Application	5
Database	5
UML Diagrams	6
Class Diagrams	6
Controller Scripts (Scripts are broken up on next page for better view)	6
Destroy Scripts	10
AI Scripts	11
Account Creation/Login & Database Communication	12
Level Scripts	12
Player Preferences & Navigational GUI	14
Animation & Audio Scripts	15
Motion Scripts	16
Animation and High Score Board Scripts	17
Animation and High Score Board Scripts	18
Overall Class Diagram	19
Design Patterns	20
Object Pooling	20
N-tier Client Server Architecture	20
Use Cases	21
Testbed	29
Programming Languages	29
Testing Tools	29
Devices Used	30
Storyboard and Paper Prototype	31
Sources Used	37
Password Hashing	37
Unity Testing Tools	37

Prefabs and Models	37
Tutorials Used & Other Prefabs	37
Music	37

2. Project Description

Universe of Things, is an arcade style game using 3D animations in a 2D environment. The plot of the game is that you are a pilot (the player) in space traveling to a distant land. The journey takes you through a vast universe where you encounter different obstacles along the way. During your journey, you fight off different space hazards and alien ships, which pose a threat to your survival. The game is not completed (won) until the player makes it to his/her home planet, which occurs after the completion of the final level.

The player will also encounter environment collectables along the way. These consist of Hearts (to collect extra lives), Stars (to collect extra lasers for your player to shoot), and just general rupees (gems to collect for future features, i.e. in app purchases with fake money). Also, many more powerups (collectables) could be added, a few that have been mentioned are missiles, and a shield. Making purchases in the application with the gems and adding the extra environment collectables (missile, shield) will occur if time permits.

The game will consist of 5 levels, each one progressively getting more difficulty to complete. Each level is not complete until the player encounters the final wave of boss hazards, whether they be alien ships or just more space trash (asteroids, etc..).

The camera is set at an orthographic perspective, giving it the look of an arcade game. Controlling the player is done in a third person perspective. The game is developed using Unity 5.5.1 and the scripting is written in C#.

Animation scenes will consist of different camera perspectives with the player's ship in view performing some type of aerial maneuver. The animation play time will be no longer than 60 seconds, and each level transition into an animation that will take the player to the next level respectively.

Finally, there is a database feature which allows the player to save their progress throughout the game. The player is securely logged in and out of the game using an encrypted password. The only credentials related to the player are their username and password. Security is not a huge concern since we are not asking for an email address. If the player forgets their password, then they must create a new account. If there is not a good connection to the database or there is no internet connection at all, then the system will proceed independently from the database and will save all information in *PlayerPrefs* provided by the *Unity.Engine* library. In this case the feature of saving the player's state after closing the application will not exist.

3. System Analysis and Decomposition

The Universe of Things application will exist in two forms, the client side (smartphone, standalone, and WebGL) application and the database that will send and receive data about the player's preferences and current state in the game. The database shall exist as an n-tier client-server architecture with a many-to-one client side application relationship.

The overall system is composed of these two parts, the application and the database. The application will be based on many different platforms. Unity 5 makes this easy, we will target Android Smartphones with O.S Android 4.0.3 (Ice Cream Sandwich) and above, iPhones with standard IOS, Linux, Windows, Mac, and aslow WebGL. The WebGL applications will run the slowest and will only be compatible with Internet Explorer, Chrome, and Safari.

3.1. Application

The application side consists of 16 different classes at the moment and they are as follows: *CoRoutines* (connection with database), *ShipSelection* (selecting players ship), *GameController* (controls the game), *Levels* (controls the flow of levels 2-5), *LevelScript_01* (controls the flow of level 1), *PickUpScript* (allows for environment collectables), *PlayerController* (controls player's movement and actions), *Boundary* (keeps the objects in the game view), *BGScroller* (adds movement to the background), *Mover* (adds movement to the hazards), *RandomRotator* (add angular rotation to space trash), *WeaponController* (AI for enemy weaponry), *EvasiveManuever* (AI for enemy attack), *DestroyByContact* (destroying objects when they are hit by another), *DestroyByBoundary* (destroying objects as the hit they leave the boundary), *DestroyByTime* (destroying objects that are transparent to the user), *GUIAccount* (login and create account user interfaces), and *GUIMenu* (navigation menu/pause menu).

As you can see many of the classes have potential for combining into one class. For example, *DestroyByContact*, *DestroyByTime*, and *DestroyByBoundary* could all be combined into one single class. *Mover* and *RandomRotator* could also be combined into one class.

3.2. Database

The second part of the system is the database used to store data about the player. The first call to the database when the application is launched is a call from *Linus()* (CoRoutine for checking connection) and a response from *Lucy()* (the PHP script that checks for a connection). If *Lucy()* says there is no connection or the connection is poor then the database will not be accessed and gameplay will proceed without it. The system will *Linus()* will not call *Lucy()* again until the application is relaunched. When *Lucy()* says the connection exists and is good then database is used.

The data sent and received will consist of the player's username and password, used to login to the database. The password will be encrypted using a salted hash. There will be no need for extra security, the database will not have any users personal information i.e. email, home address, phone number. Additional data stored about the player consists of the following: Points (collected while destroying objects), Rupees (collected as objects to pickup in environment),

Ship (the current ship the player has chosen), Level (the current level the player is playing), and Lives (the current amount of lives the player has left).

Updating the database takes place during login, gameplay and level completion. During login/create account the database is accessed and queries the full record related to the username and password received. This query is then split up into an array and distributed to the HUD (Heads Up Display/GUI) by the GameController.

Gameplay updates occur on an interval set by the programmer. We don't want to update the database after every point scored or every rupee collected, this would be a lot of forward communication for the database. The intervals for points is set to every 50 points, update the database. Same with the rupees, but the interval is every 40 (rupee value). Each rupee has a different value ranging from 1 to 20, so once the player's rupee value reaches an interval of 40, the database is updated.

Finally, the database updates the full record related to the username and password every time a level is completed. This way when the user completes a level and decides to close the application, when they log back in they will be at the next level respectively. If the user has completed the game, then they will start once again from the beginning. We hope to add the option of selecting a level from a list of levels, this would take place in the navigation menu and will only be implemented if time permits.

“Class diagrams start on next page...”

UML Diagrams

3.3. Class Diagrams

3.4. Controller Scripts (Scripts are broken up on next page for better view)

ControllerScripts

PlayerController

```
+tilt : float
+speed : float
+Boundary : Boundary
+shot : GameObject
+shotSpawns : Transform[]
-numberOfSpawns : int
+fireRate : float
-nextFire : float
+missile : GameObject
+missileSpawn : Transform
+missileCooldown : float
-gameController : GameController
-nextMissile : float
-missileShot : int
-----
+addShotSpawn() : Void
+Update() : Void
+FixedUpdate() : Void
+startToggleCollider() : Void
+toggleCollider : IEnumerator
```

GameController(Condensed)

```
-connection : int
-pc : PlayerController
-lvl : Levels
-lvl_01 : LevelScript_01
-CoRo : CoRoutines
-loadLevelWait : int
-levelCount : int
-playerDied : bool
-userName : string
-gameOver : bool
-restart : bool
-score : int
-lives : int
-rupees : int
-missileCount : int
-scoreUpdateInterval : int
-rupeeUpdateInterval : int
+shipList : GameObject[]
+rupeeBox : GameObject[]
+spawnPlayer : Transform
+numberOWavesInLevel : int
-wingDestrCnt : int
-SLH : SceneLoaderHandler
-pB : PauseNavGUI
+restartBtn : GameObject
-----
+GetItems() : IEnumerator
+SpawnRuppee() : Void
+ReSpawn() : Void
+resetLvlCount() : Void
+AddScore() : Void
+AddRuppee() : Void
+AddLife() : Void
+AddMissile() : Void
+UpdateRuppee() : Void
+UpdateScore() : Void
+UpdateLife() : Void
+UpdateMissileCount() : Void
+GameOver() : Void
+LevelCompleted() : Void
```

PickUpScript

```
-GameController : GameController
-PlayerController : PlayerController
+pickUpRuppee : AudioClip
+isAnimated : bool
+isRotating : bool
+isScaling : bool
+isFloating : bool
+rupeeValue : int
-heartValue : int
+rotationAngle : Vector3
+rotationSpeed : float
+MoveSpeed : float
+floatSpeed : float
-goingUp : bool
+floatRate : float
-floatTimer : float
+startScale : Vector3
+endScale : Vector3
-scalingUp : bool
+scaleRate : float
+scaleSpeed : float
-scaleTimer : float
-missilePickup : int
-----
Start() : Void
OnTriggerEnter() : Void
Update() : Void
```

GameController

```
-connection : int
-pc : PlayerController
-lvl : Levels
-lvl_01 : LevelScript_01
-CoRo : CoRoutines
-loadLevelWait : int
-levelCount : int
-playerDied : bool
-userName : string
-gameOver : bool
-restart : bool
-score : int
-lives : int
-rupees : int
-missileCount : int
-scoreUpdateInterval : int
-rupeeUpdateInterval : int
+currentScene : Scene
+shipList : GameObject[]
+rupeeBox : GameObject[]
+spawnPlayer : Transform
+numberOWavesInLevel : int
+restartText : GUIText
+gameOverText : GUIText
+scoreText : GUIText
+userNameText : GUIText
+livesText : GUIText
+rupeeText : GUIText
+missileText : GUIText
+wingDestrCntText : GUIText
-wingDestrCnt : int
-SLH : SceneLoaderHandler
-pB : PauseNavGUI
+restartBtn : GameObject
-----
+Start() : Void
+GetItems() : IEnumerator
+Update() : Void
+SpawnRuppee() : Void
+ReSpawn() : Void
+setGameOverText() : Void
+isGameOver() : bool
+isPlayerDead() : bool
+setRestart() : Void
+setPlayerDead() : Void
+getLvlCount() : int
+resetLvlCount() : Void
+getLoadLvlWait() : int
+getMissileCount : int
+getLivesCount() : int
+AddScore() : Void
+AddRuppee() : Void
+AddLife() : Void
+AddMissile() : Void
+UpdateRuppee() : Void
+UpdateScore() : Void
+UpdateLife() : Void
+UpdateMissileCount() : Void
+GameOver() : Void
+LevelCompleted() : Void
```

PlayerController

```
+tilt : float
+speed : float
+Boundary : Boundary
+shot : GameObject
+shotSpawns : Transform[]
-numberOfSpawns : int
+fireRate : float
-nextFire : float
+missile : GameObject
+missileSpawn : Transform
+missileCoolDown : float
-gameController : GameController
-nextMissile : float
-missileShot : int
-----
+addShotSpawn() : Void
-Update() : Void
-FixedUpdate() : Void
+startToggleCollider() : Void
-toggleCollider : IEnumerator
```

PickUpScript

```
-GameController : GameController
-PlayerController : PlayerController
+pickUpRupee : AudioClip
+isAnimated : bool
+isRotating : bool
+isScaling : bool
+isFloating : bool
+rupeeValue : int
-heartValue : int
+rotationAngle : Vector3
+rotationSpeed : float
+MoveSpeed : float
+floatSpeed : float
-goingUp : bool
+floatRate : float
-floatTimer : float
+startScale : Vector3
+endScale : Vector3
-scalingUp : bool
+scaleRate : float
+scaleSpeed : float
-scaleTimer : float
-missilePickup : int
-----
Start() : Void
OnTriggerEnter() : Void
Update() : Void
```


GameController

```
-connection : int
-pc : PlayerController
-lvl : Levels
-lvl_01 : LevelScript_01
-CoRo : CoRoutines
-loadLevelWait : int
-levelCount : int
-playerDied : bool
-userName : string
-gameOver : bool
-restart : bool
-score : int
-lives : int
-rupees : int
-missileCount : int
-scoreUpdateInterval : int
-rupeeUpdateInterval : int
+currentScene : Scene
+shipList : GameObject[]
+rupeeBox : GameObject[]
+spawnPlayer : Transform
+numberOWavesInLevel : int
+restartText : GUIText
+gameOverText : GUIText
+scoreText : GUIText
+userNameText : GUIText
+livesText : GUIText
+rupeeText : GUIText
+missileText : GUIText
+wingDestrCntText : GUIText
-wingDestrCnt : int
-SLH : SceneLoaderHandler
-pB : PauseNavGUI
+restartBtn : GameObject
-----
-Start() : Void
-GetItems() : IEnumerator
-Update() : Void
+SpawnRuppee() : Void
+ReSpawn() : Void
+setGameOverText() : Void
+isGameOver() : bool
+isPlayerDead() : bool
+setRestart() : Void
+setPlayerDead() : Void
+getLvlCount() : int
+resetLvlCount() : Void
+getLoadLvlWait() : int
+getMissileCount() : int
+getLivesCount() : int
+AddScore() : Void
+AddRuppee() : Void
+AddLife() : Void
+AddMissile() : Void
+UpdateRuppee() : Void
+UpdateScore() : Void
+UpdateLife() : Void
+UpdateMissileCount() : Void
+GameOver() : Void
```

GameController(Condensed)

```
-connection : int
-pc : PlayerController
-lvl : Levels
-lvl_01 : LevelScript_01
-CoRo : CoRoutines
-loadLevelWait : int
-levelCount : int
-playerDied : bool
-userName : string
-gameOver : bool
-restart : bool
-score : int
-lives : int
-rupees : int
-missileCount : int
-scoreUpdateInterval : int
-rupeeUpdateInterval : int
+shipList : GameObject[]
+rupeeBox : GameObject[]
+spawnPlayer : Transform
+numberOWavesInLevel : int
-wingDestrCnt : int
-SLH : SceneLoaderHandler
-pB : PauseNavGUI
+restartBtn : GameObject
-----
-GetItems() : IEnumerator
+SpawnRuppee() : Void
+ReSpawn() : Void
+resetLvlCount() : Void
+AddScore() : Void
+AddRuppee() : Void
+AddLife() : Void
+AddMissile() : Void
+UpdateRuppee() : Void
+UpdateScore() : Void
+UpdateLife() : Void
+UpdateMissileCount() : Void
+GameOver() : Void
```

3.5. Destroy Scripts

Destroy Scripts

DestroyByTime

```
+lifeTime : float  
-pB : PauseNavGUI  
-gc : GameController  
-----  
-Start() : Void  
-Update() : Void
```

Undestroyable

```
+playerExplosion : GameObject  
-gameController : GameObject  
+deadPlayer : Bool  
-----  
-Start() : Void  
-OnTriggerEnter() : Void
```

DestroyByBoundary

```
-----  
-OnTriggerExit() : Void
```

DestroyByContact

```
+explosion : GameObject  
+playerExplosion : GameObject  
+missileExplosion : GameObject  
+missileDamage : GameObject  
+spawnPlayer : Transform  
+scoreValue : int  
-GameController : GameController  
+deadPlayer : bool  
-----  
-Start() : Void  
-OnTriggerEnter() : Void
```

3.6. AI Scripts

AI Scripts

WeaponController

+shot : GameObject
+shotSpawn : Transform
+fireRate : float
+delay : float

-Start() : Void
-Fire() : Void

WeaponControllerWingmen

+shot : GameObject
+shotSpawn : Transform
+shotRate : Float

-Start() : Void
-Fire() : Void

EvasiveManuever

+Boundary : Boundary
+tilt : float
+dodge : float
+smoothing : float
+startWait : Vector2
+maneuverTime : Vector2
+maneuverWait : Vector2
-currentSpeed : float
-targetManeuver : float

-Start() : Void
-Evade() : Void
-FixedUpdate() : Void

WingmenNavigation

+playerExplosion : GameObject
- navMeshAgent : NavMeshAgent
-player : GameObject

-Start() : Void
-Move() : Void

3.7. Account Creation/Login & Database Communication

Account Login/Creation & Database Communication Scripts

Login

```
-connection : int  
-CoRo : CoRoutines  
-ELm : Text  
+username : GameObject  
+Regsiter : GameObject  
+CreateAccountBtn: GameObject  
+password : GameObject  
-----  
-Start() : Void  
-Update() : Void  
+LoginBtn() : Void  
+ExitBtn() : Void  
-StopEditorPlayback() : Void  
+setELm() : Void  
-LoadPlayerSelection() : Void
```

CoRoutines

```
+items : string[]  
-L : Login  
-SLH : SceneLoaderHandler  
-SLHo : GameObject  
-----  
-Start() : Void  
-GetDataValue() : Void  
+UpdateShipSelection() : Void  
+GetData() : Void  
+CreateAccount() : Void  
+LoginAccount() : Void  
+UpdateData() : Void  
-UpdateShipSelectionCo() : IEnumerator  
-GetDataCo() : IEnumerator  
-CreateAccountCo() : IEnumerator  
-LoginAccountCo() : IEnumerator  
-UpdateDataCo() : IEnumerator  
-setConnStatus() : Void
```

Register

```
-connection : int  
-CoRo : CoRoutines  
+ERm : Text  
+username : GameObject  
+cUsername : GameObject  
+password : GameObject  
+cPassword : GameObject  
+Login : GameObject  
-USERNAME : String  
-C_USERNAME : String  
-PASSWORD : String  
-C_PASSWORD : String  
-specialChars : String[]  
-outputString : String  
-----  
-Start() : Void  
-Update() : Void  
+SubmitBtn() : Void  
-checkPassword() : Bool  
-checkUserName() : Bool  
+BackBtn() : Void  
+setERM() : Void
```

3.8.

3.9.

3.10.

3.11.

3.12.

3.13. Level Scripts

Level Scripts

LevelScript_01

```
-gc : GameController
-SLH : SceneLoaderHandler
-pB : PauseNavGUI
+spawnValues : Vector3
+hazardCount : int
-spawnWaveCount : int
+powerUpBox_01 : GameObject[]
+synthHazards_01 : GameObject[]
+snareHazards_01 : GameObject[]
+HHHazards_01 : GameObject[]
+KDthHazards_01 : GameObject[]
+BassHazards_01 : GameObject[]
+KeysHazards_01 : GameObject[]
+SSHazards_01 : GameObject[]
-WAITING_4KEYS : bool
-RESTART_THE_BEAT : bool
-WAITING_4SOLO : bool
-WAITING_4SNARE_HH : bool
-WAITING_4BEATDROP : bool
-NEED_RESPAWN : bool
-SPAWN_POWERUP : bool
+snynthBPM : float
+snareBPM : float
+HHBPM : float
+KDBPM : float
+BassBPM : float
+KeyBPM : float
+SSBPM : float
-rythmnCount : int
-lastTime : float
-deltaTime : float
-timer : float
-----
-Start() : Void
-Update() : Void
+StartLvlOne() : Void
-ReSpawnLvl_01() : IEnumerator
-SpawnPowerUp() : IEnumerator
+checkPlayerProgress : bool
-SynthWaves() : IEnumerator
-SnareWaves() : IEnumerator
-HHWaves() : IEnumerator
-KDWaves() : IEnumerator
-BassWaves() : IEnumerator
-KeyWaves() : IEnumerator
-SSWaves() : IEnumerator
-LoadNewLvl() : IEnumerator
```

Levels

```
-SLH : SceneLoaderHandler
-pB : PauseNavGUI
-gc : GameController
+hazards : GameObject[]
+spawnValues : Vector3
+hazardCount : int
+spawnWait : float
+startWait : float
+waveWait : float
+numOfWavesInLvl : int
-spawnWaveCount : int
-beginBossWaveGeneric : bool
+bossHazardCount : int
-----
-Start() : Void
-Update() : Void
+startGenericLvl : Void
+checkPlayerProgress() : bool
+enteringBossWave : Void
-SpawnWaves() : IEnumerator
-SpawnBossWaveGeneric() : IEnumerator
-LoadNewLvl() : IEnumerator
```

LevelScript_05

```
-SLH : SceneLoaderHandler
-pB : PauseNavGUI
-gc : GameController
+hazards : GameObject[]
+FinalEnemy : GameObject
+spawnValues : Vector3
+hazardCount : int
+spawnWait : float
+startWait : float
+waveWait : float
+numOfWavesInLvl : int
-spawnWaveCount : int
-beginBossFight : bool
+bossHazardCount : int
-----
-Start() : Void
-Update() : Void
+startGenericLvl : Void
+checkPlayerProgress() : bool
+enteringBossWave : Void
-SpawnWaves() : IEnumerator
-SpawnBossLvl05() : IEnumerator
-LoadNewLvl() : IEnumerator
```

3.14. Player Preferences & Navigational GUI

Player Preferences & Navigation GUI

ShipSelection

```
-index : int  
+shipList : GameObject[]  
-userName : string  
-nextScene : int  
-CoRo : CoRoutines  
-SLHo : GameObject  
-SLH : SceneLoaderHandler  
-connection : int  
-----  
-Start() : Void  
-ToggleLeft() : Void  
-ToggleRight() : Void  
-ConfirmButton() : Void
```

LevelSelectionHandler

```
-CoRo : CoRoutines  
-SLH : SceneLoaderHandler  
-LevelTextList : GameObject[]  
-userName : String  
-connection : int  
-levelIndex : int  
-----  
-Start() : Void  
-ToggleLeft() : Void  
-ToggleRight() : Void  
-ConfirmButton() : Void
```

PauseNavGUI

```
+exitBtn : GameObject  
+ShipSelBtn : GameObject  
+resumeBtn : GameObject  
+muteAudio : GameObject  
+pauseBtn : GameObject  
+NavMenu : GameObject  
+AudioMenu : GameObject  
+PAUSE_BTN_ENABLED : Bool  
-GAME_IS_PAUSED : Bool  
+AUDIO_MUTED : Bool  
-gc : GameController  
-SLH : SceneLoaderHandler  
-CoRo : CoRoutines  
-LOOKING_4_GAMEOVER : Bool  
-userName : String  
-LEFT_SCENE : Bool  
-Menu : GameObject  
-----  
-Start() : Void  
-Update() : Void  
+ResumeBtn : Void  
+ExitBtn() : Void  
-StopEditorPlayback() : Void  
+MuteAudio() : Void  
+AudioMenuBtn() : Void  
+BackBtn() : Void  
+NavPause() : Void  
+ShipSelectionBtn() : Void  
+LevelSelectionBtn() : Void  
+GameIsPaused() : Bool  
-ActivatePauseBtn : IEnumerator  
+isLeft_SCENE() : Bool  
+setLEFT_SCENE : Void
```

3.15. Animation & Audio Scripts

Animation & Audio Scripts

PlayVideoClip

-r : Renderer
-movie : MovieTexture

-Start() : Void
-Update() : Void

MixerLevels

+AudioMixer : mainMix
+Explosions : Slider
+Blasters : Slider
+Music : Slider
+Collectables : Slider
+MasterVolume : Slider
-pB : PauseNavGUI
-WaitingForMute : Bool
+muteAudio : GameObject

-Start() : Void
-Update() : Void
-SliderValuesToMute() : Void
+SetMasterLvl() : Void
+SetMusicLvl() : Void
+SetPickUpsLvl() : Void
+SetLazerShotLvl() : Void
+SetExplosionsLvl() : Void
-ClearAll() : Void

SceneLoaderHandler

-loadingText : Text
-ImageCanvas : GameObject
-LoadingImage : Image
-loadScene : Bool
-ENTERING_NEW_SCENE : Bool
-scene : int
-flashSpeed : Float

-Start() : Void
-Update() : Void
+LoadNewSceneString() : Void
+LoadNewSceneInt() : Void
-LoadNewSceneIntCoRoutine() : IEnumerator

3.16. Motion Scripts

Motion Scripts

Boundary

+xMin : float
+xMax : float
+zMin : float
+zMax : float

BGScroller

+scrollSpeed : float
+tileSizeZ : float
-StartPosition : Vector3
-Start() : Void
-Update() : Void

Mover

+speed : float
-pB : PauseNavGUI
-gc : GameController
-lvl_01 : LevelScript_01
+isGoinToPulse : Bool
+pulseMaterial : Material
+isScaling : Bool
+startScale : Vector3
+endScale : Vector3
-scalingUp : Bool
+scaleSpeed : Float
+scaleRate : Float
-scaleTimer : Float
-childModel : gameObject
-thisRender : MeshRenderer
-thisMaterial : Material
-scaleCount : int
-Start() : Void
-Update() : Void

ShipRotationScript

-Update() : Void

RandomRotator

+tumble : float
-pB : PauseNavGUI
-gc : GameController
-Start() : Void
-Update() : Void

CometTrailScript

+TrailComet : GameObject
-Start() : Void
-waitStart() : IEnumerator
-trailWait() : IEnumerator

CometMovementScript

+speed : Float
+horizontal_speed : Float
+tumble : Float
+boundary : Boundary
-Start() : Void
-FixedUpdate() : Void

3.17. Animation and High Score Board Scripts

Animation and High Score Board

AnimationScript

```
-curScene: Scene  
-aniPlay: Animator  
-pCon: PlayerController  
-shipList: GameObject[]  
-bound: GameObject  
+playExit: boolean  
-----  
-Awake(): Void  
-Start(): Void  
+playExitAni(): Void  
-Update(): Void  
-EnableControls(): IEnumerator
```

PlayerScoreListHandler

```
+entry: GameObject  
-man: ScoreManager  
-lastChangeCounter: int  
-----  
-Start(): Void  
-Update(): Void
```

ScoreManager

```
-changeCounter: int  
-playerScores: Dictionary<string,int>  
-records: String[]  
-items: String[]  
-username: String[]  
-points: int[]  
-rupees: int[]  
-lives: int[]  
-----  
-Init(): Void  
+GetScore(): int  
+SetScore(): Void  
+ChangeScore(): Void  
+GetPlayerNames(): String[]  
+GetChangeCounter(): int  
-GetDataCo: IEnumerator  
-GetDataValues: String  
-OnEnable(): Void
```

3.18. Boss AI and Health Scripts

Boss AI and Health

Level_4_Boss_AI

```
+Big_shot: GameObject
+Big_shotDLt: GameObject
+Big_shotDR: GameObject
+small_shot: GameObject
+ShotSpawn: Transform
+ShotSpawn_Left: Transform
+ShotSpawn_LeftM: Transform
+ShotSpawn_LM: Transform
+ShotSpawn_Right: Transform
+ShotSpawn_RightM: Transform
+ShotSpawn_RM: Transform
+fireRate: float
+delay: float
-PercentHealth: float
-attackChoic: int
-----
-Start(): Void
-delayFire(): Void
-attack(): Void
-DoubleCross(): IEnumerator
-CrissCross(): IEnumerator
-RightStrike(): IEnumerator
-LeftStrike(): IEnumerator
-LeftM_RightM_M_Fire(): IEnumerator
-LeftM_RightM_Fire(): IEnumerator
-LM_RM_Fire(): IEnumerator
-Fire(): IEnumerator
```

Level_4_Boss_Health

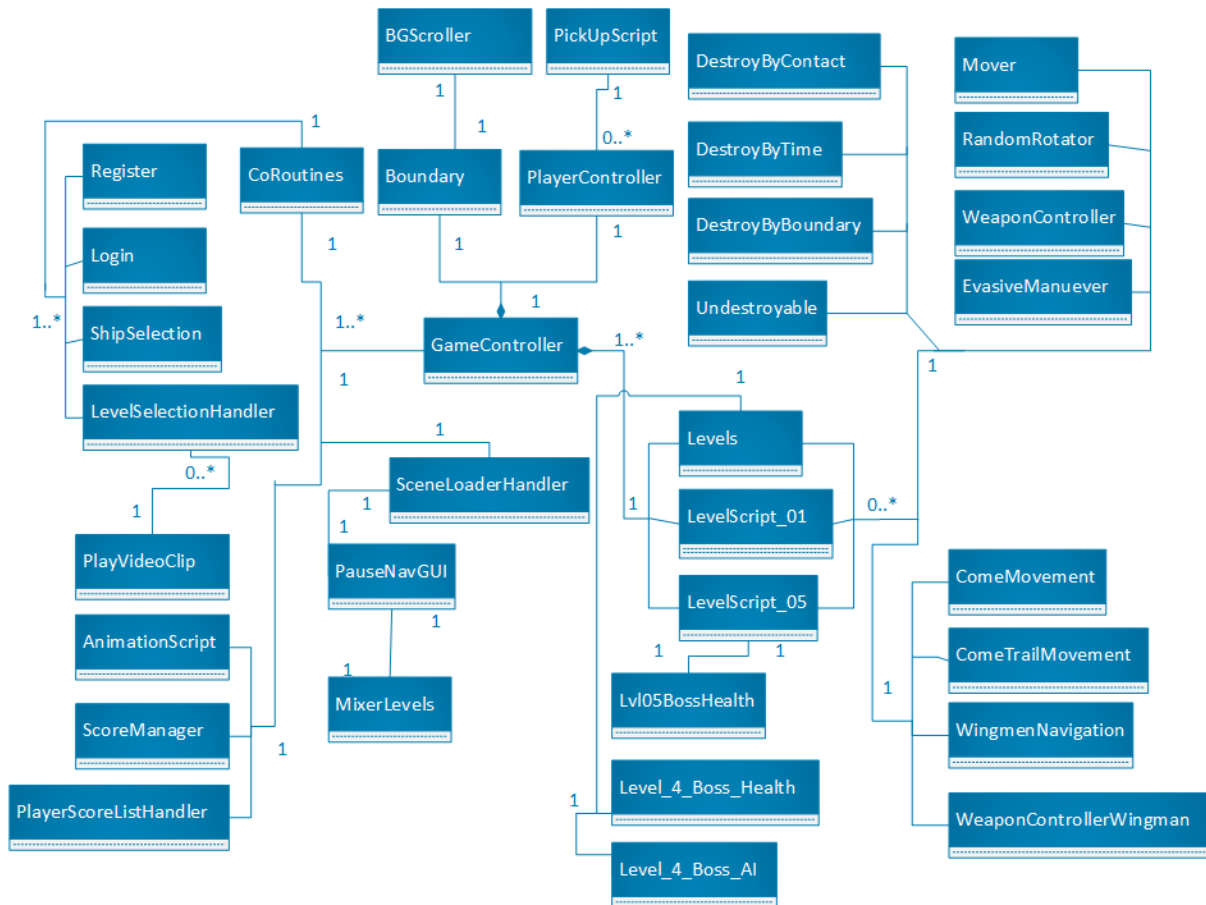
```
+StartHealth: float
+CurrentHealth: float
+explosion: GameObject
-----
-OnTriggerEnter(): Void
```

Lvl05 BossHealth

```
+Health: int
-gc: GameController
+missileExplosion: GameObject
+explosion: GameObject
+enemy: GameObject
-hasSpawned1: boolean
-hasSpawned2: boolean
-hasSpawned3: boolean
-----
-Start(): Void
-OnTriggerEnter(): Void
-Update(): Void
+getHealth(): int
```

3.19. Overall Class Diagram

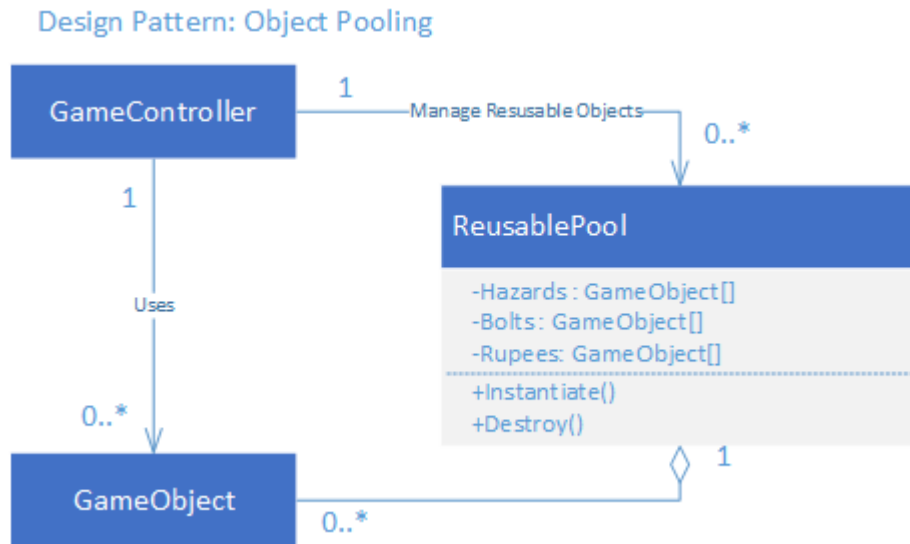
Class Diagram



3.20. Design Patterns

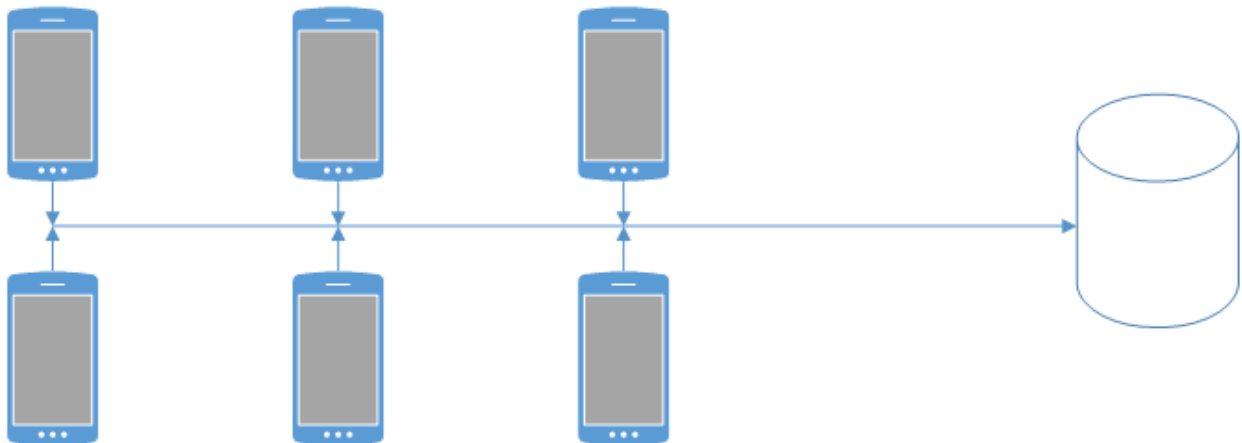
3.20.1. Object Pooling

The architectural design pattern for the application side of the system is an object pooling pattern. The game controller manages the reusable pool of objects instantiating them into the scene as well as destroying them after they have left the view.



3.20.2. N-tier Client Server Architecture

The architectural design for the database is an N-tier many to one smartphone to database relationship. The application will receive information from the database and the database will update all clients with respective data.



3.21. Use Cases

Use Cases:

1. Launching Application
2. Creating an account
3. Logging in
4. Selecting ship
5. Selecting level
6. Moving player
7. Shooting laser
8. Collecting rupee
9. Pausing game
10. Changing Audio preferences
11. Restarting the level

Name: Launching the application

Actor: User

Priority: Optional

Pre-Conditions: An instance of the application has not been started

Post-Conditions: An instance of the application starts

Steps

Actor actions	System Response
1. The user starts the application	2. The system creates an instance of the application and loads the main menu

Alternative

1. The user starts the application	2. The system resumes a previous instance of the game to the pause screen
------------------------------------	---

Name: Creating an account

Actor: User

Priority: Optional

Pre-Conditions: An instance of the application is running.

Postconditions: An account is created for the player

Steps

Actor actions	System Response
1. The user clicks 'Create Account' on initial screen	2. The system loads the 'Create Account' GUI with 'username', 'confirm username', 'password', 'confirm password' text fields
3. The user enters an acceptable username, confirms it, an acceptable password and confirms it	4. The system checks if the strings are acceptable and match, and then checks if there is a matching username in database. If not, the account is created

Alternative

1. The user clicks 'Create Account' on initial screen	2. The system loads the 'Create Account' GUI with 'username', 'confirm username', 'password', 'confirm password' text fields
3. The user enters an unacceptable username, confirms it, an acceptable password and confirms it	4. The system shows an error to the respective error in the username/password string

Alternative

1. The user clicks 'Create Account' on initial screen	2. The system loads the 'Create Account' GUI with 'username', 'confirm username', 'password', 'confirm password' text fields
3. The user enters an acceptable username, confirms it, an acceptable password and confirms it	4. The system system checks if the strings match, the username is already part of the database
	5. The system shows an error that the account already exists

Name: Logging into the game

Actor: User

Priority: Optional

Pre-Conditions: An instance of the application is running.

Postconditions: The player logs into the game

Steps

Actor actions	System Response
1. The user enters credentials into the username and password text fields	2. The system checks the credentials against the database containing user information
	3. The system loads the saved values of spaceship type, levels completed, and high scores.

Alternative

1. The user enters credentials into the username and password text fields	2. The system checks the credentials against the database containing user information
	3. The system returns an incorrect credentials flag
	4. The system clears the text field awaiting user input

Alternative

1. There is no database connect	2. The system loads a new instance of a player profile to play in offline mode
---------------------------------	--

Name: Selecting a ship

Actor: User

Priority: Required

Pre-Conditions: An instance of the application is running. The user has logged in

Postconditions: The ship is selected

Steps

Actor actions	System Response
1. The user clicks the left or right arrow to traverse the ship list	2. The system renders the respective ship every time the left or right arrow is clicked
3. The preferred ship is highlighted, the user clicks the 'Select' button	4. The system saves the preference in the database

Name: Selecting a Level

Actor: User

Priority: Required

Pre-Conditions: An instance of the application is running, the user is logged in and has selected a ship

Postconditions: The game loads the selected level

Steps

Actor actions	System Response
1. The user clicks the left or right arrow to traverse the level list	2. The system renders the respective level every time the left or right arrow is clicked
3. The preferred level is highlighted, the user clicks the 'Select' button	4. The system saves the preference in the database

Name: Moving the spaceship

Actor: User

Priority: Optional

Pre-Conditions: The user is inside a loaded level

Post-Conditions: The spaceship shoots a laser

Steps

Actor actions	System Response
1. The user presses a directional key	2. The system checks if the spaceship instance is on the border of the moveable boundary side with respect to the key being pushed
	3. If the spaceship is not on the border, the system moves the spaceship to in the appropriate direction respective to the key
	4. The system moves the spaceship until the key is no longer pressed or the border of the moveable boundary has been reached

Alternative

1. The user presses a directional key	2. The system checks if the spaceship instance is on the border of the moveable boundary side with respect to the key being pushed
	3. If the spaceship is on the border, the system does not move the spaceship in any direction

Alternative

1. The user moves the joystick (mobile)	2. The system checks if the spaceship instance is on the border of the moveable boundary side with respect to the joystick direction
	3. The system moves the spaceship until the key is no longer pressed or the border of the moveable boundary has been reached

Alternative

1. The user presses a directional key	2. The system checks if the spaceship instance is on the border of the moveable boundary side with respect to the joystick direction
	3. If the spaceship is on the border, the system does not move the spaceship in any direction

Name: Shooting lasers

Actor: User

Priority: Optional

Pre-Conditions: The user is inside a loaded level

Post-Conditions: The spaceship shoots a laser

Steps

Actor actions	System Response
1. The user clicks the left mouse button	2. The system creates a new instance of the laser at the ship's location
	3. The system renders the laser movement from the location of the spaceship upon click
	4. The system moves the laser from the initial location until the end of the screen or until contact with another game object with collision properties

Name: Collecting a rupee

Actor: User

Priority: Optional

Pre-Conditions: The player is inside a game, a rupee has been spawned upon destruction of an enemy game object

Postconditions: The value of the rupee type is incremented

Steps

Actor actions	System Response
1. The user collides with the spawned rupee	2. The system increments the rupee counter and increments the specific type of rupee (lives, shots, etc)
	3. The system destroys the game object

Name: Pausing the game

Actor: User

Priority: Optional

Pre-Conditions: The game is running. The game is not paused.

Postconditions: The game pauses

Steps

Actor actions	System Response
1. The user clicks the pause button	2. The system changes the state of the game to paused
	3. The system saves the values of all existing variables and asset instances
	4. The system will not allow any other game functions to be performed

Alternative

1. The user closes the application while a game has been created and a level has been initialized	2. The system changes the state of the game to paused
	3. The system saves the values of all existing variables and asset instances
	4. The system will not allow any other game functions to be performed

Alternative

1. The user presses the pause button	2. The system changes the state of the game to running
	3. The system resumes all functions included in the gameplay

Alternative

1. The user opens the application to a previous instance of the game	2. The system changes the state of the game to running
	3. The system resumes all function included in the gameplay

Name: Change audio settings

Actor: User

Priority: Optional

Pre-Conditions: The game is running. The game has been paused

Postconditions: The audio settings are changed

Steps

Actor actions	System Response
1. The user clicks the 'Audio' button from the pause menu	2. The system loads the audio interface
3. The user moves a slider to an audio setting	4. The system updates settings according to slider value
4. (Optional) The user clicks the 'Mute Audio' button	5. The system mutes all audio
6. (Optional) The user clicks 'Reset Audio'	7. The system resets the audio settings to the original values

Name: Fire Missiles

Actor: User

Priority: Optional

Pre-Conditions: The user is inside level 5

Post-Conditions: The spaceship fires a missile

Steps

Actor actions	System Response
1. The user clicks the right mouse button	2. The system creates a new instance of the missile at the ship's location
	3. The system renders the missile movement from the location of the spaceship upon click
	4. The system moves the missile from the initial location until the end of the screen or until contact with another game object with collision properties

Name: Restart Level

Actor: User

Priority: Optional

Pre-Conditions: The user died and has 0 lives

Postconditions: The player respawns in the level

Steps

Actor actions	System Response
1. The user presses the 'Restart' button on screen	2. The system resets stats and starts the player at the beginning of the current level

4. Testbed

4.1. Programming Languages

Programming Languages used in this project consist of mainly C# and a little bit of JavaScript to script the logic of the game. The IDE used is MonoDevelop or Visual Studio depending on if you have a PC or MAC. MonoDevelop for MAC and Visual Studio for PC.

There is an addition use of PHP scripting for the database communication. These scripts are upload to the server and are independent of the logic of the game. The only store or retrieve data related to the player's preferences, score, and progress.

The builds will consist of stand alone builds for PC and Mac, WebGL builds for browser playing, and Mobile builds for Android and IOS.

4.2. Testing Tools

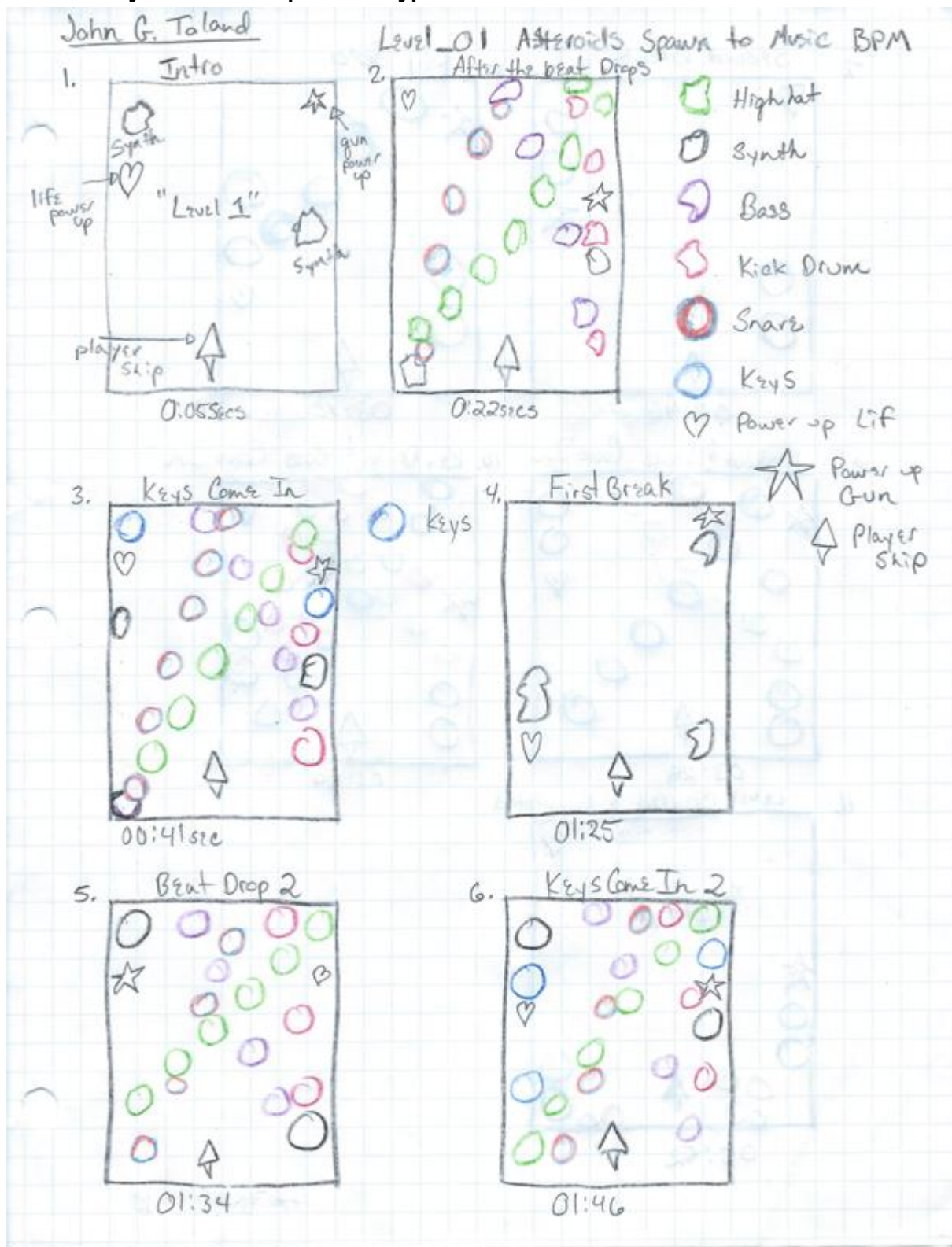
The tools used for testing are provided by the *Unity Community* via the *Asset Store*. The package is called *Unity Testing Tools* and the link is provided in section **4.5 Sources Used**. The testing suite provides Integration, Assertions, and Unit testing. There will be a scene in the project files call *TestScene*, this is where all the tests will be implemented and ran.

4.3. Devices Used

Devices used are personal computers for development and testing. Android and IOS smartphones for testing.

The programming takes place in the Unity Editor, Unity 5.0 and above. The scripting IDE used are either MonoDevelop or Visual Studio as mentioned above.

4.4. Storyboard and Paper Prototype



7. Second Break



02:46

8. Key Solo



03:10

9. Highhat & Snare Come In



03:29

10. Key Drum & Bass Come In

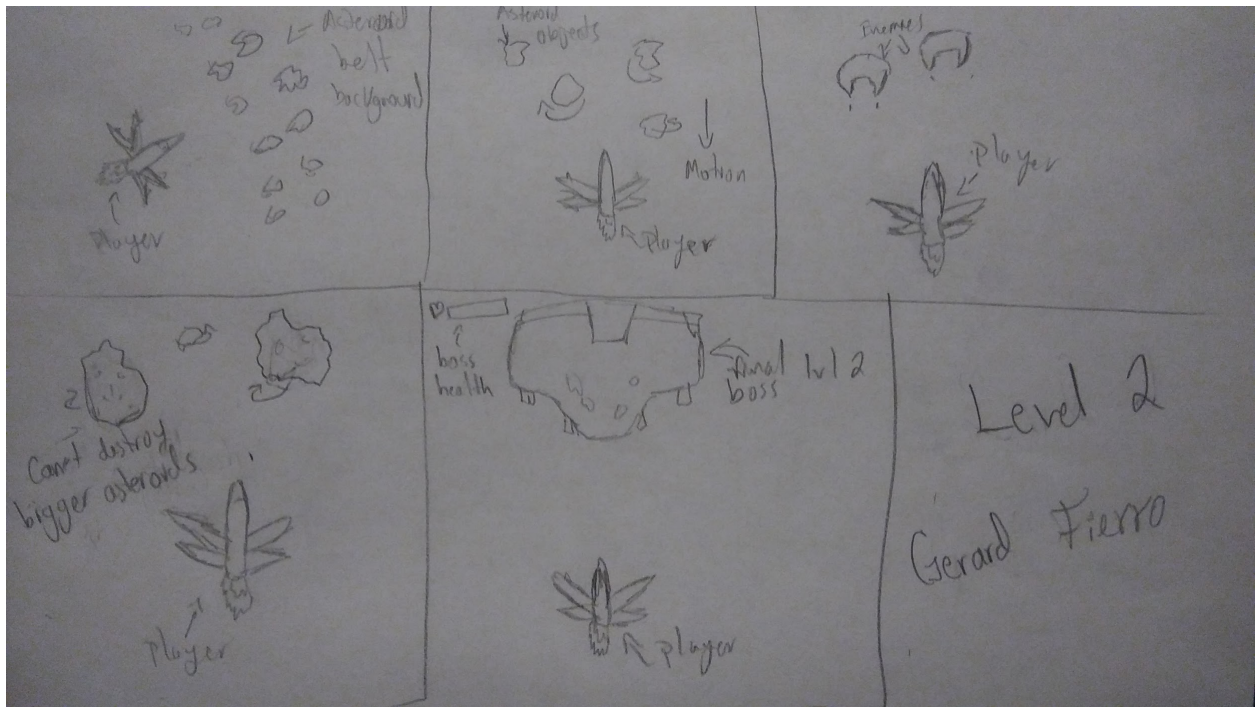


03:50

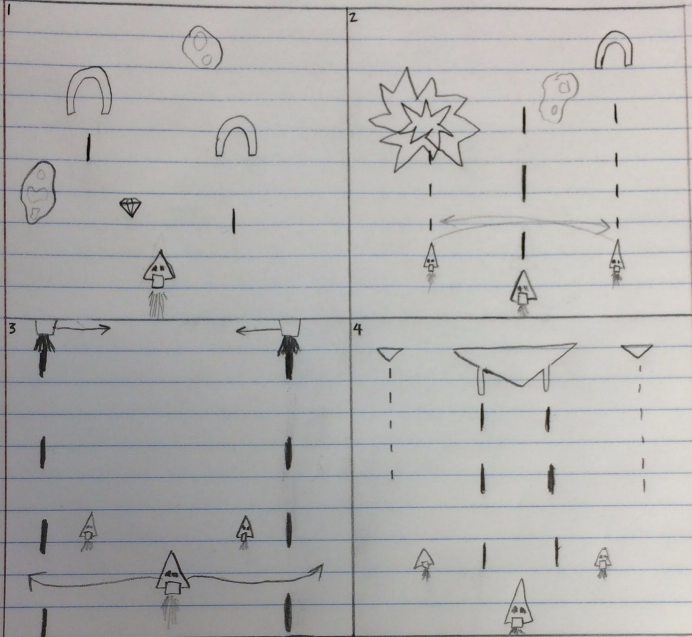
11. Level Cleared & Completed



05:52

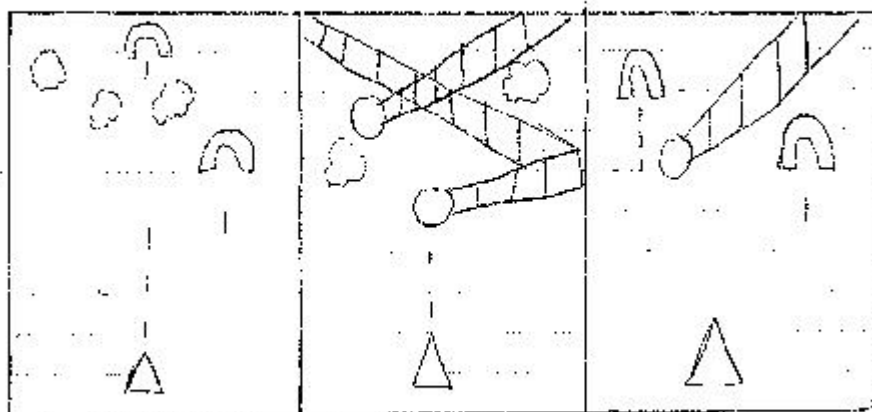


Level 3

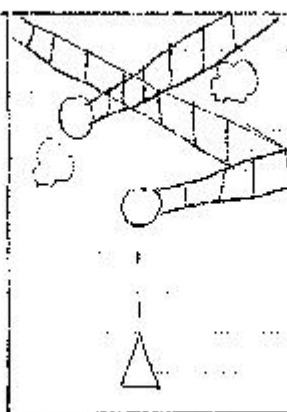


1. Asteroids and enemies. Pickup for wingmen.
2. Wingmen move independently from user's ship. Shoot constantly.
3. Cannon's appear from left and right, move to opposite sides of screen
4. Boss with 2 wingmen

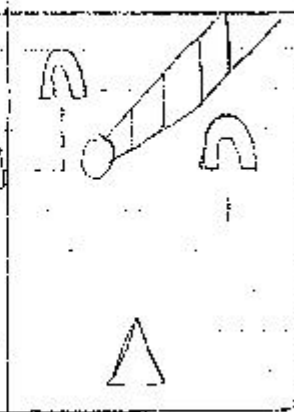
Level 4



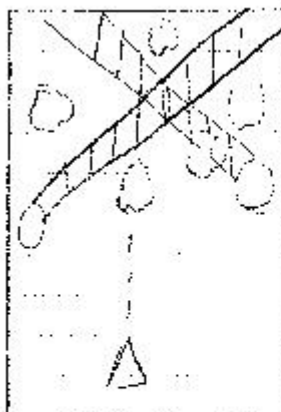
Wave 1
meteors &
enemies



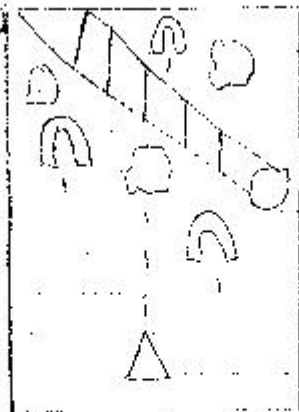
Wave 2
ice comets &
meteors



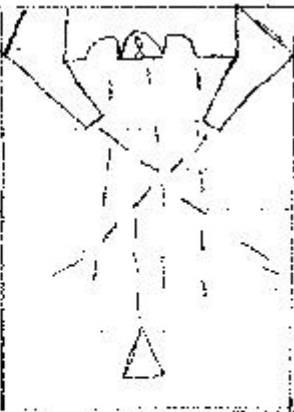
Wave 3
enemy ships &
ice comets



Wave 4
ice comets &
meteors

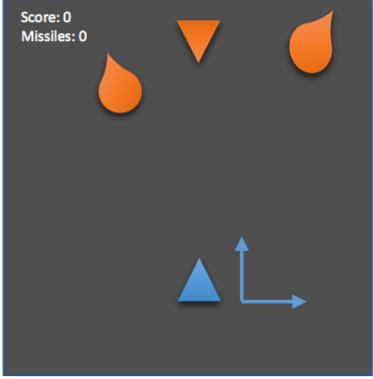
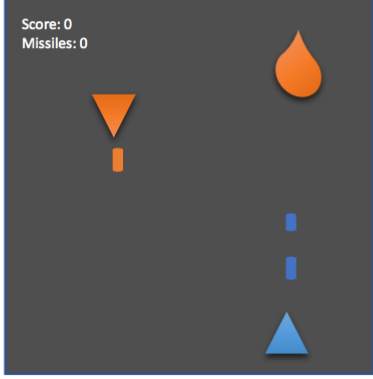
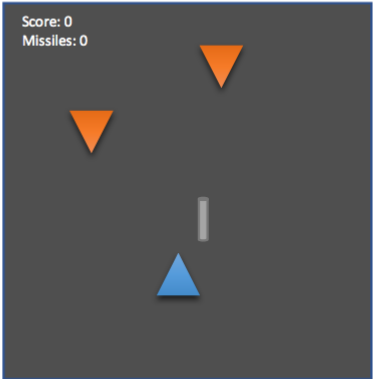
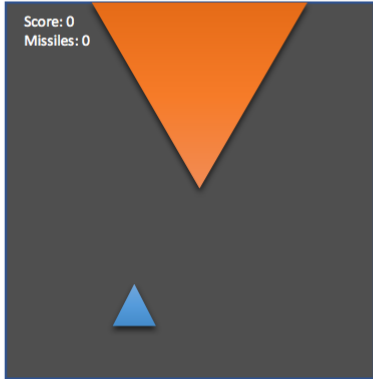


Wave 5
enemy ships,
ice comets &
meteors



Wave 6
Boss

Level_05 Paper Prototype

 <p>The player has free movement over the X and Y axis, and will be presented with hazards such as asteroids and enemy A.I. to destroy for score, or maneuver out of the way to survive.</p>	 <p>If the player meets a hazard or is shot by the enemy's weapon, then he is deducted a life. If a hazard meets the player's weapon, they will be destroyed and added to score.</p>
 <p>Items are available for pick-up by the player making contact with the item.</p>	 <p>The level will conclude with a Boss for the player to defeat. The boss will have a health pool and after the health pool is empty the player will have won</p>

4.5. Sources Used

4.5.1. Password Hashing

Hashing passwords in PHP link:

<http://php.net/manual/en/faq.passwords.php>

4.5.2. Unity Testing Tools

Tutorial link:

<https://unity3d.com/learn/tutorials/topics/production/unity-test-tools>

Unity Testing Tools Asset Store link:

<https://www.assetstore.unity3d.com/en/#!/content/13802>

4.5.3. Prefabs and Models

Rupee Models link:

<https://www.assetstore.unity3d.com/en/#!/content/73764>

Space Ships Models link:

<https://www.assetstore.unity3d.com/en/#!/content/73167>

<https://www.assetstore.unity3d.com/en/#!/content/46128>

<https://www.assetstore.unity3d.com/en/#!/content/4392>

<https://www.assetstore.unity3d.com/en/#!/content/11711>

Planets/Skyboxes:

<https://www.assetstore.unity3d.com/en/#!/content/3392>

<https://www.assetstore.unity3d.com/en/#!/content/56841>

4.5.4. Tutorials Used & Other Prefabs

SpaceShooter Tutorial link:

<https://unity3d.com/learn/tutorials/projects/space-shooter-tutorial>

4.5.5. Music

The music listed below is used in the game and is not royalty free. As of right now we have not paid any royalties to use this music. We are only using this music for means of educational purposes. If we take the application to market or intent to do so, we will seek licensing the use of the songs with ASCAP, American Society of Composers, Authors, and Publishers.

www.ascap.com/licensing

Login Menu:

Artist: MF Doom

Song: Monosodium Glutamate

Album: Special Herbs and Spices Vol. 7 & 8

Year: 2004

Player Selection:

Artist: MF Doom

Song: Camphor

Album: Special Herbs and Spices Vol. 1

Year: 2005

Level Selection:

Artist: Bluetech

Song: Honey in the Heart

Album: The Divine Invasion

Year: 2009

Level_01:

Artist: Bluetech

Song: Holding Space

Album: The Divine Invasion

Year: 2009

Level_02:

Artist: Bluetech

Song: Phoenix Rising

Album: The Divine Invasion

Year: 2008

Level_03:

Artist: Bluetech

Song: Pitch Black-Ape to Angel (Bluetech Remix)

Album: Sines and Singularities

Year: 2005

Level_04:

Artist: Bluetech

Song: Oleander

Album: Elementary Particles

Year: 2009

Level_05:

Royalty Free Music