## Post-mortem report

Overall I am pleased with my development of this assignment project. The resulting product resembles and plays similarly to the original Space Invaders and I am proud of this. Planning my development cycle and time management early on was crucial in my development of the game and I found it to be very helpful. I used a mixture of waterfall and iterative style design methodology when writing my code. To begin with I planned what gameplay functionality needed to be implemented and in what order. The rough order of implementation I designed was: main menu, state system, sprite rendering, player and alien movement, collisions, score and finally boost tasks such as player health, barriers, audio and bonus motherships. Starting development early on meant I had time near to the end to implement some of the additional features that I had planned but not all of them. I am pleased with my class organisation and believe the number of classes as well as their organisation fits the project fairly well.

On the other side, I do believe there is plenty of room for improvement in my coding. Firstly, due to time constraints I was not able to adhere to the OOP style of coding as much as I would have liked. Most of my projects functionality resides in the Game class rather than placing mechanic implementation in more relevant class files. This means each class is heavily coupled to the Game class and some code is not encapsulated entirely in relevant classes and this was not what I wanted or designed initially. Furthermore, some parts like the collision functions could definitely be condensed and combined to provide more efficient code. On another note, I ran into some problems whilst coding the project. One problem I encountered was the differing performance when playing on more or less powerful computers; this was due to the fact that some mechanics such as movement ran off of the update tick of the gameplay which was obviously faster on high powered computers. To get around this I implemented a tick counter on the game loop which I then used mostly when sprites needed to be moved as this meant the movement was determined by the speed of the PC and so did not change as much when changing computer. My implementation solves this for the most part but left some issues such as the mothership occasionally lagging and the player movement differing depending on direction.

A final area of improvement would be the additional features I would have liked to implement had I more time. This would include functionality such as: enemy movement based on enemies left as well as enemy height, barrier sprites being destroyed depending on where they were shot, additional player lives past 3, different difficulty levels, a high score system, additional audio, sprite animation and finally some power ups. The power ups would probably have been activated by score or a high multiplier and this could have included: enemies moving and shooting in slow motion, player machine gun mode and enemies moving in random directions.

## Requirement planning

## UML diagrams

|  |
| --- |
| **Game** |
| * Invader : ASGE:Sprite * Player\_one : ASGE:Sprite * Bullet\_one : ASGE:Sprite * Mothership\_one : ASGE:Sprite * Aliens : vector<Enemy> * Bullets : vector<Bullet> * Barriers : vector<Barrier> * Lives : vector<Player> * Explosion : ASGE:Sprite * Left : ASGE:Sprite * Right: ASGE:Sprite * LeletterP : ASGE:Sprite * Escape : ASGE:Sprite * Space : ASGE:Sprite * One : ASGE:Sprite * Two : ASGE:Sprite * Three : ASGE:Sprite * Game\_state : GameState * State\_callback\_id : int * Game\_callback\_id : int * Time\_difference : float * Death\_counter : float * Mothership\_spawn\_timer : float * Alien \_move\_counter : float * Alien\_move\_speed : float * Alarm\_counter : float * Alien\_shoot\_speed : float * Move\_id : int * Audio\_engine : irrKlang::ISoundengine |
| * proccessGameActions : void * Input : void * Run : virtual bool * shouldExit : bool * Render : void * Init : virtual bool * drawFrame : virtual void * initAudio : const bool * playAlarm : void * checkPlayerAlive : const void * deathDelay : void * loadEnemies : void * moveAliens : void * renderAliens : const void * enemyShoot : void * changeAlienSpeed : void * checkAlienLives : const void * resetEnemies : void * loadBarriers : void * renderBarriers : const void * resetBarriers : const void * changeBarriers : const void * renderMothership : const void * deployMothership : void * checkMothershipCollision : void * updateGame : void * updateMenu : const void * updateOptions : const void * updatePause : const void * updateGameOver : const void * resetGame : void * loadUI : void * renderUI : void * loadControls : const void * renderMenuUI : const void |

|  |
| --- |
| **GameActor** |
| * Health : int * Is\_alive : bool |
| * getHealth : const int * getAlive : const bool * setAlive : void * setHealth : void |

|  |
| --- |
| **Mothership** |
| * Mothership : ASGE:Sprite |
| * getXPosition : const virtual int * getYPosition : const virtual int * loadEnemy : virtual void |

|  |
| --- |
| **Player** |
| * Score : int * Multiplier : int * Death : bool * Player : ASGE:Sprite |
| * movePlayer : void * loadPlayer : void * getXPosition : const int * getYPosition : const int * getScore : const int * setScore : void * getMultiplier : const int * setMultiplier : void * setDeath : void * getDeath : const bool |

|  |
| --- |
| **Barrier** |
| * Barrier : ASGE:Sprite |
| * loadBarrier : void * changeBarrier : void * resetBarrier : void * getYPosition : const int * getXPosition : const int |

|  |
| --- |
| **Bullet** |
| * Missed : bool * Bullet : ASGE:Sprite |
| * loadBullet: void * moveBullet : void * setMissed : void * setBullet : void * getMissed : const bool * getXPosition const int * getYPosition : const int |

|  |
| --- |
| **Enemy** |
| * Direction : float * Start\_y : float * Can\_shoot : bool * Enemy : ASGE:Sprite |
| * setCanShoot : void * setDirection : void * setStartY : void * getDirection : const int * getCanShoot : const bol * getXPosition : const virtual int * getYPosition : const virtual int * getStartY : const int * loadEnemy : virtual void |

## Class diagram

## 

## Pseudo-code

**loadSprite (player, enemy, barrier etc)**

Create sprite with renderer

Set sprite x position

Set sprite y position

Set sprite scale

Set sprite texture

**setBullet**

Set bullet at x coordinate of firing point

Set bullet at y coordinate of firing point

Set bullet to alive

**moveBullet**

If bullet sprite is off screen

Kill bullet

Set missed to true

If bullet is alive

Move bullet on y axis by value passed in

Else

Hide bullet off screen with - x and y coordinates

**loadBullets**

For every enemy bullet

Add to vector

Vector item ->loadSprite

**renderBullets**

For every enemy bullet

bullet->loadSprite

If game state == playing

moveBullet

**loadEnemies**

For 5 rows

For 11 columns

Add to vector

enemy->loadSprite

enemy->set x position

enemy->set y position

enemy->set direction

If enemy is on top 2 rows

Change sprite

Adjust position to match others

If enemy on bottom row

Set can shoot to true

Increment vector and x position

Reset x position back to start

Increment y position

**moveAliens**

Increment movement tick counter

If movement tick > move speed float

For every alien

If alien is alive

If alien is out of bounds on x axis

Set change to true

If change is true

For every alien

If alien is below y threshold

Kill player / end game

Move alien down y axis

If alien is alive

If alien is out of bound on x axis

Move back in bounds

Reverse alien direction

Else

For every alien

Move along x axis

Reset movement tick counter

**renderAliens**

For every alien

If alien is alive

Render with renderer

**enemyShoot**

For every alien

If alien is not on bottom row

Is alien below could shoot but is dead

Set alien able to shoot

If alien can shoot

For every enemy bullet

Random number between 0 and enemy shoot speed variable

If random number is 1

If bullet is not already alive

If alien is alive

Set bullet at point of alien

Play shooting sound

**changeAlienSpeed**

If aliens are between lower y bounds

Increase movement speed

Increase shooting frequency

If aliens are below even lower y bounds

Increase movement speed

Increase shooting frequency

If aliens are near bottom of y axis

Increase movement speed

Increase shooting frequency

**checkAlienLives**

For every alien

If alien is alive

Return

If player health < 3

Increase by 1

If alien starting y position < certain y point low down

For every alien

Increment starting y position

resetEnemies

**resetEnemies**

Get new starting y position

For 5 rows

For 11 columns

Set alien to alive

Reset x position

Reset y position

Reset direction

Reset can shoot

If on bottom row

Set can shoot

Increment vector and x position counter

Reset x position

Increment y position

**renderBarriers**

For every barrier

If barrier is alive

Render barrier sprite

**resetBarriers**

For every barrier

Set to alive

Reset health to 3

**changeBarriers**

For every barrier

Set texture depending on health of barrier

Else if health != 1, 2 or 3

Kill barrier

**renderMothership**

If mothership is alive

Render mothership

**deployMothership**

If mothership is alive

Increment mothership spawn timer with tick

If mothership spawn timer is above threshold

Set mothership to alive

Reset mothership x position

Reset mothership spawn timer

Else if mothership is alive

If mothership is past x axis threshold

Kill mothership

Return

Move mothership along x axis

playAlarm

**playAlarm**

If mothership is alive

Increment alarm counter with tick

If alarm counter is past threshold

Play alarm sound

Reset alarm counter

**checkPlayerAlive**

If player is not alive

Set game state to game over

Else

If player is not in death state

Render player

**deathDelay**

If player is set to death state

Increment death counter with tick

If death counter is past threshold

Set player to no longer in death state

Reset death counter

**checkCollision**

If player bullet is alive

For every alien

If alien is alive

If alien x position overlaps bullet x position

If alien y position overlaps bullet y position

spawnExplosion

Kill alien

Play explosion sound

Kill bullet

Increase player score depending on alien and multiplier

**checkBarrierCollision**

For every barrier

If barrier is alive

For every alien bullet

If alien bullet is alive

If barrier x position overlaps alien bullet x position

If barrier y position overlaps alien bullet y position

spawnExplosion

Play explosion sound

Kill bullet

Reduce barrier health by 1

If barrier health == 0

Kill barrier

**checkPlayerCollision**

For every enemy bullet

If bullet is alive

If player x position overlaps alien bullet x position

If player y position overlaps alien bullet y position

spawnExplosion

Play explosion sound

Kill bullet

Reduce player health by 1

Set player death state to true

Reset multiplier

Reset x position

If player health == 0

Kill player

**spawnExplosion**

Set explosion coordinates to coordinates passed in

**resetGame**

resetEnemies

resetBarriers

Reset alien movement speed

For every alien

Reset starting y position

Set player to alive

Set player health to 3

Set score to 0

Reset player x position

For every enemy bullet

Kill bullet

Kill player bullet

**loadUI**

For every life sprite

Add to vector

loadSprite

Set x and y position

Increment x position

loadSprite for keyboard button sprites

**renderUI**

Render return text

Render escape button sprite

Render score text

Conver player score to text

Render player score

Render multiplier text

Convert player multiplier to text

Render player multiplier

Render lives text

For every player life

Render life sprite

**loadControls**

Render return text

Render escape button sprite

Render pause text

Render p button sprite

Render shoot text

Render space button sprite

Render move text

Render a button sprite

Render d button sprite

**renderMenuUI**

Render 1 button sprite

Render 2 button sprite

Render 3 button sprite

**processGameActions**

If game action == exit

Game state = exit

If game action == play

Game state = play

If game action == controls

Game state = controls

If game state== pause

If game action == pause

Game state = playing

Game action = none

If game state == playing

If game action == pause

Game state = pause

Game action = none

If game action == right

If player x position is below maximum

movePlayer

If game action == left

If player x position is above minimum

Move player (negative direction)

If game action == shoot

Game action == none

If player bullet is not already alive

setBullet at player position

Play shooting sound

**Input**

If game state == game over

If key == escape

If key == pressed

Game action = return

If key == released

Game action = none

If game state == controls

If key == escape

If key == pressed

Game action = return

If key == released

Game action = none

If game state == main menu

If key == escape

If key == pressed

Game action = exit

If key == released

Game action = none

If key == 1

If key == pressed

resetGame

Game action = play

If key == released

Game action = none

If key == 2

If key == pressed

Game action = controls

If key == released

Game action = none

If key == 3

If key == pressed

Game action = exit

If key == released

Game action = none

If game state == pause

If key == p

If key == pressed

Game state= playing

If key == released

Game action = none

If game state == playing

If key == escape

If key == pressed

Game action = exit

If key == released

Game action = none

If key == p

If key == pressed

Game state = pause

If key == released

Game action = none

If key == space

If key == pressed

If player is not in death state

Game action = shoot

If key == released

Game action = none

If key == A

If key == pressed

If player is not in death state

Game action = left

If key == released

Game action = none

If key == D

If key == pressed

If player is not in death state

Game action = right

If key == released

Game action = none

**updateGame**

deathDelay

checkPlayerAlive

renderUI

changeBarriers

renderBarriers

Render player bullet

renderAliens

moveAliens

renderMothership

deployMothership

changeAlienSpeed

Move player bullet

If player bullet missed

Reset multiplier

Set bullet to missed

enemyShoot

checkCollision

checkPlayerCollision

checkBarrierCollision

checkMothershipCollision

Render explosion

Hide explosion

renderBullets

proccessGameActions

**updateMenu**

Render menu text

renderMenuUI

**updateOptions**

Render controls text

loadControls

**updatePause**

renderUI

renderBarriers

checkPlayerAlive

Render player bullet

Render explosion

renderAliens

renderMothership

renderBullets

Render pause text

proccessGameActions

**updateGameOver**

renderUI

renderBarriers

checkPlayerAlive

Render player bullet

Render explosion

renderAliens

renderMothership

renderBullets

Render game over text

proccessGameActions