**SCHOOL OF COMPUTING**

**UNIVERSITY OF TEESSIDE**

**MIDDLESBROUGH**

**TS1 3BA**

**Games Engine Construction (COM2063)**

**ICA Assignment 2  
Game Engine**

**Michael Thomas**

**S6099298**

**10/01/2018**

Table of Contents

[1 User Guide 3](#_Toc503032114)

[2 Maintenance Guide 4](#_Toc503032115)

[3 Conclusion 5](#_Toc503032116)

[4 REFERENCES/BIBLIOGRAPHY 6](#_Toc503032117)

# User Guide

## How to play the demo

The game was tested in OL9 on Lab Machine – TU46944

Intergalactic War is a space shooter style game, where you must survive as long as you can against the endless waves of enemies that come in all directions. You start with 3 lives that are displayed on screen and once you lose all your lives the game is over. You take damage when an enemy collides with you, to prevent taking damage you can fire bullets and evade them as best as you can. Pickups at 5 points to your score and killing an enemy scores you 1 point.

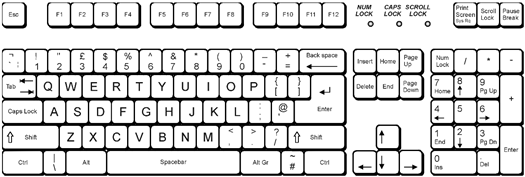
Pause/Resume

Move player left/right

Shoot (Right Trigger)



Restarts the game



Shoot

Move Right

Move Backwards

Move Left

Move Forward

Pause/Resume

## Know issues / bugs in this release

The current know issues/ bugs in this release are that occasionally as the scrolling background moves it flickers this maybe something to do with the tickTime which dictates after how many milliseconds can the game be updated e.g. key press, player moves etc. Another issue is pickups overlapping each other and sometimes when the game starts pickups spawn on top of the player so the player gets 5 points instantly. Lastly the enemies also overlap each other too, which doesn’t make it look very good.

## Implementation Check list

**Adequate Rated Items (D to C)**

**Graphics**

* A ‘black boxed’ graphic system is in place
* Textures can be efficiently drawn to arbitrary positions on and partially off the screen (clipped)
* Animation is implemented and working correctly

**World State**

* A player entity exists
* Input is recognised and can be used to alter the world state e.g. move the player entity
* The Xbox controller is supported and ‘hot pluggable’

**Code Quality**

* Class interfaces are minimal and complete. Class function and member variable visibility is correct
* Code can be built and executed without compiler errors or warnings in debug and release
* Code is well commented
* There are no memory leaks
* There is good error handling throughout
* You have followed all the submission requirements e.g. made a video, submitted the correct files etc.

**Report**

* All requested sections have been attempted adequately and the report is professionally presented

**Good Rated Items (C to B)**

**World State**

* A world model system is in place. It is separate from other code and black boxed
* There is a game loop handling input, world update and rendering
* Bounding rectangle collisions are detected
* There are multiple world entity types

**Code Quality**

* Good use of object oriented techniques e.g. polymorphism, member variable visibility
* Memory is only allocated / deallocated outside of the game loop
* Const is used correctly

**AI**

* Some AI routines are in place e.g. enemy entities move around the world following paths, use state machines etc.

**Report**

* This report would allow another programmer to work with your code systems

**Other**

* Some sound effects are in place

**Excellent Rated Items (B to A)**

**Graphics**

* Interpolation is used to smooth entity movement

**World State**

* The player entity can shoot projectiles (or equivalent functionality)
* Explosion and bullet management
* Game play is independent of platform capabilities (i.e. uses a model tick approach)
* Game cycling e.g. detection of win / lose conditions and restarting the game
* There is a scoring system with the score shown on screen

**AI**

* Several different enemies with differing behaviours

**Report**

* This report has insightful and balanced reflection

**Extra Marks (Examples)**

* Mapping of world space on to screen space
* Other graphics techniques have been implemented e.g. background scrolling, blending modes etc.
* Level data is loaded from a file
* A difficulty level
* More advanced C++ e.g. use of namespaces, STL, C++ 11 and further patterns
* A\* algorithm
* ‘Intelligent’ enemy behaviour
* There are sound effects for collisions, explosions and firing
* Additional black box systems have been implemented e.g. for AI, Sound
* HUD features beyond simple text e.g. health bars, mini maps etc.
* Other features, please list below:

# Maintenance Guide

## Game Systems (Visualisation, World Model)

### Visualisation System

Firstly, the purpose of the visualisation system is to draw textures to the screen, which I used to draw all my sprite sheets to create a visual representation of my space shooter game. This also handles sprite animation if the sprite has one. For example, I render an explosion sprite sheet that shows on the screen when the player or enemy is destroyed, so it loops through each frame to get the animation effect. The visualisation class blits the textures to the screen with the option of alpha or no alpha (faster performance if there’s no alpha).

As I already stated above I use this system in my game to render all my sprite sheets onto the screen. Since the visualisation system’s main purpose is drawing textures to the screen it is very easy to reuse this system in other games, all you would need to do is set the width, height and load the system into memory and load the textures at the start of the runtime.

The advantages of this linkage are that it allows you to reuse this system very easily to draw and render textures onto the screen with the use of unordered map you can set textures by name e.g. player, tank, enemy and it renders the correct sprite onto the screen for you. Whereas with the disadvantages of the visualisation system is that there aren’t any big disadvantages of the linkage since its purpose is to draw the textures, but you do need to make sure to delete the textures and visualisation system when the game is exited since it is on the heap. The only other real disadvantage of this linkage is that it only supports 2D rendering of sprites and would need modifying to allow 3D rendering also to draw textures in a certain position this system needs a 2D vector type to do so.

Visualisation System classes: visualisation class, sprite class

### World Model System

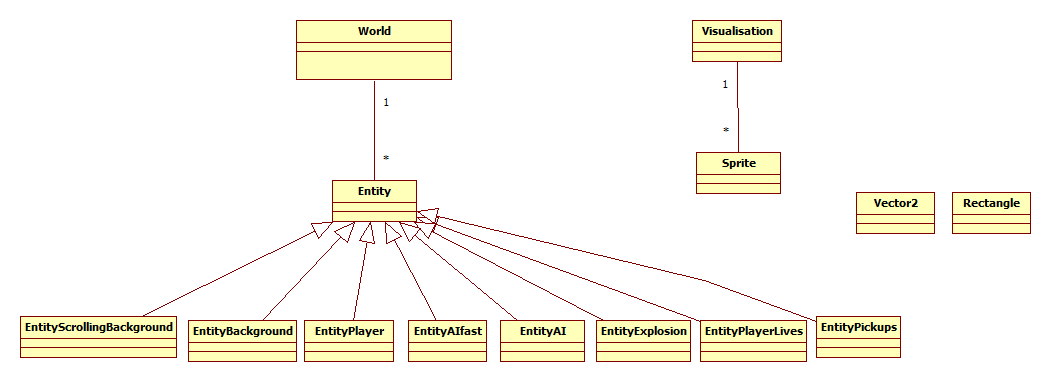
The purpose of the world model is to manage a representation of the game world and its state like calling the visualisation system to draw textures, update loop (player movement) etc. So, this system the main system where it calls all the functions so the game runs which then called from main.cpp.

The world model system is linked to my game so everything that is displayed is called from the world model whereas this system is tailed to the game it is which for me is a space shooter game and reusing this system in other games like a platformer would be hard to reuse. A lot of code would need to be changed since the controls, sprites movement etc. are completely different in the platformer than in a space shooter.

However, there are advantages of this linkage is that some parts of the system can be used in other games like update loop, loading all the data at the start of the program and drawing textures. Whereas there are disadvantages to this linkage, with the world model a lot of the system would need to be redone to suit the style of game it is and every game has different world models because every game does not play the same, there is always at least slight differences, such as using the system in a platformer you would need to remove all the other sprites, change how the player works like add gravity since the space shooter doesn’t have gravity etc.

World Model System classes: World class, Entity class, EntityAI class, EntityAIFast class, EntityBullet class, EntityExplosion class, EntityPickups class, EntityPlayer class, EntityPlayerLives class, EntityScrollingBackground class, EntityBackground class

All these classes derive on the entity class and each child class has unique behaviours that are represented in the world class such as the EntityPlayer class which updates the player position, movement etc. another example is the EntityBullet class which has collision and updates the bullet to move when the space bar or right trigger (controller) is pressed.



This UML diagram shows the hierarchical design of my space shooter game.

# Conclusion

## Critical Reflections

Looking back at the game I created there are a few things in my work I would like to reflect on, like of how I implemented the scrolling background with the flickering issue that occurred and that I created two new entities to create the scrolling background and could have probably only used one class and find a not so hard coded way of the scrolling background and getting the screen height and width, so it was more robust.

Elsewhere, I feel that the game came out good with state machines, multiple AIs, game cycling, scoring and onscreen lives. The demo is very much a fully functional game with only scoring and maybe become a bit repetitive with no enticement but with the game cycling it feels like a fully operational game with the game getting harder every time your score gets higher and higher to make it a bit more competitive.

## Lessons learned

One lesson I learned was taking on too much other little features and not focusing on the key features that needed to be implemented like AI. Also, I learned to stay with one idea which originally my space shooter game but midway creating my game engine I decided I wanted to do a platformer but didn’t realise how much of a task it was to do with collision being a big issue and with the limited time I had I decided I didn’t have enough time to get what I wanted done so I went back to my original idea of a space shooter game.

## Future enhancements

Lastly, the future enhancements I would like to implement into the game would be more advanced AI that doesn’t just follow a path but seeks you and moves towards you making it more of a challenge. I would also add more pickups like health packs, shields (block damage) and lastly a HUD system that doesn’t just show the lives and score but show a health bar, bullet cooldown bar and even a mini map of where other enemies are that when the player moves the screen moves its position and makes it feel more a world you can move around in and not just a locked screen. Plus, I would make the game cycling smoother when the game is restarted.

# REFERENCES/BIBLIOGRAPHY

<https://opengameart.org/content/stereotypical-90s-space-shooter-music> - Background music

<https://opengameart.org/content/space-shooter-art> - Space shooter sprite sheets

<http://www.classicgaming.cc/classics/space-invaders/sounds> - Sound effects

<https://opengameart.org/content/kenney-fonts> - Fonts

<https://opengameart.org/content/63-digital-sound-effects-lasers-phasers-space-etc> - Pickup sound effects

<https://www.online-tech-tips.com/computer-tips/right-click-keyboard-shortcut/> - Keyboard picture

https://www.gamespot.com/forums/system-wars-314159282/dualshock-3-versus-xbox-360-controller-25959204/ - Xbox 360 controller picture