Joint Project 1

(Programming and Graphics)

This project (game) **must implement the functionality (skills) asked for in this specification**. You should **talk** to your lecturer if your game does not, to make sure it is suitable.

The coding will be done in sections. You will be asked to demo sections of the project in the lab. We are giving you the basic design of the classes. You are to use the class structure as given in the file **Joint Project 1 Design.doc**. If you want to use a different class structure for the game, you need to talk to your programming lecture in advance of doing so. This is to make sure that your class structure will work for this game. As you code the project you may need to add some extra data members (instance variables) and functions than those given in the design document. There may be some data and function members you decide not to use.

There is a **project template** on your common drive - Programming\Labs\Joint Project 1**.** It contains the main game loop and loads the font. It also contains some of the header files you will need in the project. You **should** take a copy of it and use it to start your project.

You are being given a set of images to use in the game. These image files are to be used to represent the player and enemy sprites etc. in the game. You will find the images on your common drive in the following folder: **sprites\Joint project 1**

You are NOT allowed to use any other image files.

Your game will be marked based on **the functionality of the game**, not how it looks.

# It is absolutely prohibited to copy anyone else’s design in your project. You can ask for help with a particular problem from friends, colleagues, the lecturers in the lab etc but you must write your own design solution.

# If you get help from someone or take design/code from the web or elsewhere, you have to comment in your code to state what help you got and from where.

**Your game should contain demonstrations of the following skills:**

## Sprite movement

* move up/down/left/right
* fires weapon

**Example:**Player can move in four directions via the keyboard and is able to fire a bullet; the player disintegrates when the player is killed.

## Collision detection

* Collision detection between game entities.

**Example:**- Player dies when it comes in contact with an enemy sprite  
- Hostage destroyed when comes in contact with an enemy sprite  
- Enemy sprite dies if hit with player weapon

## Autonomous game entities

* entities that move in a straight line, then change direction every 6 seconds or entities could move left to right and right to left across the screen continuously.
* entities move up/down/left/right by themselves towards another entity
* entities die when hit by player weapon

**Example:**-Hostage moves at a constant speed for 6 seconds, then changes direction.  
-Enemy sprite moves towards Hostage.  
-Enemy sprite destroyed when hit by player’s weapon

## Display GUI

* show lives left
* show score

**Example:**Show player’s lives left on top left, and score is shown on top right.

## Gameplay

* Must have a goal/goals and rules that make it fun (difficulty must be balanced).
* The speed of your sprites in the game should **get faster** as the game progresses.
* Your game should have is the ability to start a new game when the player is killed.

**Example:**  
**Rules:**  
Game over if you lose three lives  
Win game if you kill all the enemies  
Increase score if you kill an enemy  
Decrease score if you lose a hostage  
**Goal:**  
Kill all the enemies  
**Balance:**   
the speed of the enemies and the bullet makes it possible to win but at the same time it isn’t easy.

## Game Description

You should have **two** enemy classes in the game which represent two different types of enemy. One enemy class has different behaviour to the other enemy class. For example one type of enemy could follow the player and the other could move in a straight line, then change direction every 6 seconds. Another type of enemy could move across the screen in a straight line and when it reaches the other side it should change direction and move across the screen again. You are to store one of your types of enemy in an array and the other can be stored in 2-3 individual variables or in another array if you prefer.

You don’t need to have lots of objects in this game. For example you could have one player, one bullet, 2-3 of one of the types of enemies and an array of the other type of enemy in the game. **4-6 sprites** on the screen at once is fine.

Your game should be able to regenerate the bullet and enemies. For example there should be only one player bullet on the screen at once and the enemy could come back again once killed.

Your game should have the ability to **start a new game** when the player is killed and display the game score and player lives of the previous game before the new game starts.

The speed of your sprites in the game should **get faster** as the game progresses.

**This is the basic functionality of the game.**

You must use use **Classes** and **Objects** in your Project.

**Demo:** There is a **demo** of the project which you can run to see what a project might look like. See the folder Programming\Labs\Joint Project 1\Student Demo Project on your common drive. Run the file **JointProject.exe**

## Submission Dates

**You will be asked to demo parts of your project working before the final demo. This is the minimum you should have done on these dates.**

**Monday 5th of Feb demo** the player moving around within the boundary of the screen using 4 keys.

**Monday 12th of Feb demo** the two enemy types moving around within the boundary of the screen.

**Monday 19th of Feb demo** collision detection between game entities (including player weapon fire) and scoring.

Final Project code due: **Wednesday the 21thof February (put on your M drive).**

**Final demos are taking place Monday 26th of February in the lab.**

## Submission

Submit your project to the M drive. Make sure your project is stored in a folder called **JointProject**YourName i.e. **JointProject**ThomasSmyth on your M drive. Also sign the Plagiarism Declaration (in the project folder of your common drive) and copy it to your project folder.

**Very Important:**You are required to demonstrate the project in class and to submit it to the M drive. **If you do not demonstrate the project in class you will NOT receive any marks for the project. Late entries will not be considered.**

There will be a **written exam based on the basic functionality** of the project, it will NOT include any extra functionality. You need to pass this exam in order to be allocated your demo project mark. If there is a considerable difference between your written exam mark and your demo core mark, your demo mark may be reduced downwards. You must sit the written exam in order for your project demo mark to be allocated.

**This project is worth 15-20% of your final mark in Graphics and Programming.**

## Coding

Pay attention to the programming standards (given out in class and on your common drive). To do well here, you will need:

* Make use of a 1D array of objects in the game.
* Proper structure and layout of classes & objects in your project, including correct indentation of code.
* Well commented code - used to explain pieces of code throughout functions where necessary.
* Well named, appropriate classes & variables (remember to keep your variables as local as possible.)
* Appropriate use of conditional expressions and loops
* The code should be divided into well named functions, each of which do one thing; you should be able to pass arguments to the functions and return values from them.

## Extra Functionality

As always, you will be rewarded for extras that add to the game. To achieve extra marks the added functionality must add to the game and be non-trivial.

**Extra functionality** is worth **40%** in the programming and graphics subjects.

Part of your extra functionality should be that your game has **progression:** How does the game keep the user involved, how does it get more challenging?

**Other examples of extra functionality are:**

* Introducing new enemy behaviours (enemy types) to the game. Different enemies must have different behaviours to count as extra functionality.
* Using different types of weapons (bomb, guided missile, beam etc.).
* The enemies can fire a weapon also.
* Program eye(s) that tracks an enemy.

**Marking Scheme**

|  |  |  |
| --- | --- | --- |
|  | **Games** | **Core** |
| **Correctness** | 40% | 80% |
| **Code Quality** | 20% | 20% |
| **Extra Functionality** | 40% |  |
| **Total** | **100%** | **100%** |