Game Project Extension and Extra Functionality

Due Date: Thursday Week 12 (24th May, 11:55 PM) - Weight: 25%

The Brief:

For your final assignment you will be required to build upon your assignment 2a submission by adding extra functionality to enhance the gameplay.

The Implementation:

- You must use a Version Control System such as GIT or CVS to keep track of your code. BitBucket offers free cloud repositories for individuals and small teams: https://bitbucket.org/product/pricing?tab=cloud
- Game Balancing: you must provide a way to easily experiment with variables such as enemy attributes, bullet damage, shooting frequency and bullet speed etc. to help you balance the game and make it a reasonably enjoyable experience (i.e. No 'magic numbers').

Player Controls

The player controls and camera should be updated to the following:

- The player must navigate the world using a first-person camera with standard controls (mouse to look around, WASD for movement). ✓
- The player should have no visible form. ✓
- Player movement is **no longer locked to the grid**, they can **move freely around the board** (make sure they don't fall off the edge!) ✓
- The player should be able to shoot bullets which will do a random amount of damage between an upper and lower bound if they hit an enemy. ✓

Health Packs/Tiles

Health kit

Health Packs/Tiles will appear as in assignment 2a but with the following difference:

- Collision detection must be done using the collision system in the game engine used. ✓

Enemies

As with assignment 2a there should be 5 enemies of different strengths.

- Enemies are represented as humanoid meshes (provided in the Assets zip file) ✓
- Enemies must always face the player, so when the player moves, enemies will rotate to face the player. ✓

Enemy Battle

- The Enemy Battle logic is to be replaced with a shooting mechanic where the enemies will shoot at the player ✓
- Each enemy has a random amount of health calculated between a lower and upper bound ✓
- Enemy bullets will deal a random amount of damage calculated between a lower and upper bound ✓
- Enemies will shoot at the player at random intervals
- Bullets will travel at a constant speed ✓

Enemy Movement

The enemies should exhibit the following movement:

- Stronger enemies should move slower, while weaker enemies move faster.
- The first enemy should constantly chase the player. ✓
- The second enemy should continually move away from the player. ✓
- The third enemy should continually move to random points on the board. ✓
- The fourth enemy should try to head the player off by moving to a point somewhere in front of the player (the distance is up to you) ✓
- The fifth should remain still, but when the player gets too close, they should flee to a random point.
- If an enemy catches the player, or the player collides with an enemy, the player is killed instantly ✓
- The enemy mesh is removed from the board after it is defeated in battle. ✓

Sprites and Text

No change from Assignment 2a. Basic usage of sprites and text will be required to convey information to the player including: ✓

- The player's health should be visible using a health bar sprite ✓
- Text should be used to show the players score. ✓

Collisions

All collisions must be done using the game engine collision system. ✓

Additional Functionality (20%)

The final part of this assignment requires you to add some additional functionality of your own choosing. Examples of additional functionality may include but are not limited to:

- Sound effects and ambient sound.
- Experimentation with custom shaders.
- Exploration of procedural content and level generation.
- Enhanced gameplay elements including additional pickups, a more elaborate health system, a difficulty and level-up system, in-game currency with purchasing, and the ability to save high scores.
- Code refactoring/cleanup and implementation of design patterns (factory, memory pools etc.)
- Anything else you can think of! Personal exploration of concepts not formally taught in this unit are welcome.

Week 11 Beta Testing:

- It's important to seek feedback from others as developers are often too close to a project and issues with the gameplay can sometimes go unnoticed. We'll be conducting a beta test in the week eleven tutorial where you'll observe as your peers play your game. You'll also be providing feedback to others.
- Your tutors will play-test your game and will be awarding a maximum of 10 marks based on how far you've progressed since assignment 2a. Your game doesn't have to be finished by this point, however a playable build must be presented with noticeable additions from the base assignment 2a submission.
- The beta test will be marked in class – no Moodle submission is required for this component. Please be ready to demo upon arrival to class.

Week 12 Final Submission:

A readme.txt file must be included in your final week twelve submission which lists all additional functionality included and any usage instructions (extra controls for instance) and a link to your online code repository.

Your final assignment must be uploaded to Moodle before the due date. Any late assignments will be penalised 5% of the total available mark per day late.

Students seeking extensions will have to follow the Special Consideration guidelines outlined here: http://www.monash.edu.au/exams/special-consideration.html (note: you will need to provide adequate documentation to support your claim for an extension).

Please include an assignment coversheet with your submission. They can be found here: http://infotech.monash.edu.au/resources/student/forms/

Week 11 Beta Test – 10%					
	No		Yes		
Does the game compile and initialise successfully? (Hurdle) If not, you will receive a mark of 0 for the beta test.			✓		
	N	P	C	D	HD
▪ Has sufficient progress been made since assignment 2a?			[✓]		

	No		Yes		
Does the game compile and initialise successfully? (Hurdle) If not, the maximum mark you can receive for this assessment will be capped at 50 out of a possible 100 marks.			✓		
Game Functionality – 70%					
	N	P	C	D	HD
Online code repository and version control system used? (05)					✓
Are the values for game variables stored as constants in a header file? (05)					✓
Are the player movement and controls implemented correctly? (05)					✓
Do the AI characters behave as expected? (20)					✓
Do the player and AI shoot in a convincing way (bullet spawning at end of gun etc.). (05)					✓
Is the collision detection implemented correctly? (10)					✓

Additional Functionality? (20)		[illegible]	[illegible]		
Code Quality – 20%					
	N	P	C	D	HD
Is the code written in an efficient manner? (including good memory management) (5)					✓
Does the code exhibit good object-oriented design? (10)					✓
Is the readability and style of the code to a high standard? Are **appropriate comments** included throughout? (5)					✓