GAME2005 – Game Physics

Assignment 3
2D Collision Detection and Response

Due: week #12 (Friday November 27, 2019) @ midnight.

Value: 15%

Maximum Mark: 100

Overview:

This Assignment is broken down into two parts. Each part should be hosted in a separate Scene.

The best way to complete the assignment is to complete as much of the work as possible on your own. Then get together with your group members to compare what you have and share ideas, handing in the best assignment possible.

Programming Required:

This is a programming assignment. You will be required to use the SDL Template provided (SDL Engine). Please **do not** create your own template (e.g. a project that does not use the code base demonstrated in class). Completing this assignment will help you with the Final Test.

Project Assets:

You will be responsible to acquire (or create) both image (textures) and sound assets for the project.

Instructions :

Base: Project Setup and UI

1. You will use **C++ and the SDL Framework Provided (SDL Engine)** to create two scenes (20 Marks: GUI, 25 Marks).
    a) Your application should include a **Start Scene** with **labels** that show Team members' names and student IDs. Include two Buttons or other controls which will allow the user to go to either **Scene 1** or **Scene 2** (5 Marks: GUI)
    b) **Scene 1** will contain the Physics Simulation for **Part 1** (Simple 2D Collision Detection) and should include appropriate assets for the Physics Simulation as indicated in **Part 1**. You should include a background image or Tile-Map (5 Marks: GUI)

c) **Scene 2** will contain the Physics Simulation for **Part 2** (Simple 2D Collision Detection and Response) and should include appropriate assets for the Physics Simulation as indicated in **Part 2**. You should include a background image or Tile-Map (5 Marks: GUI)
d) Choose an **appropriate scale** for each scene (i.e. pixels per meter – PPM). Add appropriate variables and **data structures** (5 Marks: Functionality)
e) Let the user "activate" each scene with a button or other control (5 Marks: Functionality)
f) Use appropriate **label objects** in your scene to show **key statistics** which may include mass, position, velocity, acceleration, force, etc. for each scene (you may use ImGui for this as it has been included with your template). (5 Marks: GUI, 5 Marks: Functionality)
g) Allow the User to **change the variables** for each Scene. Update your Display Accordingly. (5 Marks: Functionality)
h) Provide the User with a button or other control within each scene to go back to the Start Scene (5 Marks: Functionality).

Part 1: Simple 2D Collision Detection

1. Using the **SDL Template** provided (SDL Engine), create a simple Scene where it is raining bullets. (20 Marks: Functionality)
    a) There should be at least 10 bullets on the screen at any one time. In fact, you will only ever have 10 bullets in the game. All bullets will be reused (Hint: create a Bullet Pool). (5 Marks: Functionality)
    b) The bullets will initially move down the screen at a rate of gravity. The bullets will appear at the top of the screen at random locations across the x-axis. Bullets will reset offscreen when they pass the bottom border of the screen. (5 Marks: Functionality)
    c) The player will control a sprite that has an appealing texture (a spaceship, a plane, etc.). You may use mouse motion or keyboard input to control the player. (5 Marks: Functionality)
    d) If the player's ship gets hit by any bullet an explosion sound clip will play. Ensure that the sound clip only plays once per bullet. (5 Marks: Functionality)

Part 2: Simple 2D Collision Detection and Response

2. Using the **SDL Template** provided (SDL Engine), create a simple Scene that includes a bouncing ball and a player controlled "brick". Use the appropriate collision detection and response techniques to solve this problem (20 Marks: Functionality)
    a) The Bouncing ball will collide with the scene's boundary and maintain most of its **momentum**. However, some of its energy will be lost and it will continue to slow down. Select an appropriate mass for the Bouncing Ball. Allow the user to change the shape of the ball to a square, triangle, or other polygon and re-rerun the simulation (10 Marks: Functionality).
    b) The player can move a brick on the screen using mouse controls. When the brick collides with the bouncing ball, the bouncing ball will shift direction and potentially gain momentum depending on the velocity with which it is hit by the player's brick. (10 Marks: Functionality).

Part 3: Video Demonstration

2. Create a Short Video presentation with your favourite screen capture and streaming tool (OBS Recommended) and upload it to Blackboard. You must also include a short PowerPoint (or Google Slides) **Slide Deck** that includes a **single slide** to start your video (10 Marks: Video Demo)
    a) The first (and only) Slide of your Slide Deck must include a **current image** of you and your partner(s) (no avatars allowed) that is displayed appropriately on the page. You must also include your **Full Name(s)**, **Student ID(s)**, the **Course Code**, **Course Name**, and your **Assignment information**. (2 Marks: video)
    b) You will demonstrate each of your app's **Scenes**. Your UI must be **clearly visible** (2 Marks: Video)
    c) You will describe the **code** and **functionality** of your application (2 Marks Video).
    d) **Sound** for your Video must at an appropriate level so that your voice may **be clearly heard**. Your Screen should be **clearly visible** (2 Marks: Video).
    e) Your Short Video should run no more than 5 minutes (2 Marks: Video).

Part 4: Version Control

3. Share your files on **GitHub** to demonstrate Version Control Best Practices
    a) **Create** an appropriately named repository on GitHub (1 Marks: Version Control)
    b) Your repository must include your code and be well structured (2 Marks: Version Control).
    c) Your repository must include commits that demonstrates the project being updated at different stages of development – each time a major change is implemented (2 Marks: Version Control).

Note: Your project will not be accepted without your video demo

Evaluation Criteria

Feature	Description	Marks
Physics Simulation UI	UI Controls meet the application requirements. Display elements are deployed in an attractive manner. Appropriate contrast is applied to application UI Controls and any background colours applied so that all text is legible.	20
Physics Simulation Functionality	Your Simulation works without errors. Controls are available for the user to modify the initial values and restart the simulation with those values.	65
Version Control	GitHub commit history demonstrating regular updates.	5
Video Presentation	Your short video must demonstrate your app working in the simulator and discuss each of your designs	10
Total		**100**

SUBMITTING YOUR WORK

Your submission should include:

1. Include the Name and StudentID of each team member at the top of your document.
2. A link to your working GitHub repository for your simulation
3. A zip archive of your project uploaded to Blackboard
4. A link to your Video Demonstration.

This assignment is weighted **15%** of your total mark for this course.

Late submissions: 20% deducted for each additional day late.