German University in Cairo
Media Engineering and Technology
DMET 502 Computer Graphics, Winter 2024
Assoc. Prof. Dr. Rimon Elias

Assignment 1
Deadline: Friday 18th of October
Individual Assignment

Description & Game Play:

General idea:

In this assignment, you are required to implement a 2D infinite runner side-view game of your own design. The main models in the game; the player, obstacles that the player should dodge, collectables and power ups that the player can collect. The main idea of the game is that the player dodge any obstacle in the scene and collects as many collectables as possible within the game time.

The player can either jump or duck while staying within the game boundaries and avoiding obstacles. As time progresses, the game speed ramps up, and the player should aim to gather as many collectables and power-ups as they can.

At the top of the screen, the player can see their health, score, and remaining game time. They have five lives represented by a health bar or shapes rather than numbers. Each collision with an obstacle results in the loss of a life. The game ends if the player loses all their lives or if the time runs out. The score increases as the player collects items throughout the game.

Modeling:

Primitive types include shapes as: point, line, line strip, triangle, quad, and Polygon (can be found in Lab 1). Some models would include different primitives which means that you can not use the same primitive type twice.

- The **upper and lower game boundaries** should have **at least 4** primitives each excluding the time, score and health.
- The **player** is drawn with **at least 4 different** primitives.
- The **obstacles** are drawn with **at least 2 different** primitives, and appear infinitely throughout the game at varying heights.
- The **collectables** are drawn with **at least 3 different** primitives, and appear infinitely throughout the game.
- The **2 different powerups** are drawn with **at least 4** primitives each.
- The two types of powerups, collectables and obstacles should appear **different than each other.**
- Health should be drawn with **at least 2** primitives at the top of the screen.
- All objects in the game **should not** overlap in the scene.

German University in Cairo
Media Engineering and Technology
DMET 502 Computer Graphics, Winter 2024
Assoc. Prof. Dr. Rimon Elias

The requirements:

- **The environment:**
  - Health, Score, and Time are displayed on the top of the screen.
  - Health should be displayed non-numerically and gets updated upon live lost.
  - The player.
  - Collectables and power ups.
  - The game speed increases over time, causing all objects to reach the player faster as time progresses.
  - Upper and lower borders.
- **Player:**
  - The player starts initially on the left side of the screen and standing on the ground.
  - Motion is controlled by keyboard function; jump and duck,
  - Can not move outside the game borders,
  - The player loses one of the **five** lives upon colliding with any of the obstacles and the health bar gets updated.
- **Obstacles:**
  - The obstacles appears from the right side of the screen,
  - moving towards the player's position,
  - The obstacles should be positioned on ground **and** slightly above the player's height, allowing the player to dodge by ducking.
  - Obstacles **continuously spawn** throughout the duration of the game.
- **Collectables:**
  - The collectables **disappear** upon collision with the player,
  - Upon collection, the game score **increases**.
  - Collectables **continuously spawn** throughout the duration of the game.
  - The player should be able to **miss** the collectables, for example, if the collectable appears above the player and the player did not jump (did not collide with it), the player **does not acquire** the collectable.
- **Power-ups:**
  - You are free to choose the functionality of the power up, however you should have **two** different power ups functionalities.
  - At least **two of each type** appears every game.
  - The powerup should **disappear** if acquired by the player.
  - The effect of the power-up should last for a few seconds then **deactivates**, hence you can not use a functionality that does not deactivate by time, such as health gain or increase game time.
  - The player should be able to **miss** the power up, for example, if the power-up appears above the player and the player did not jump (did not collide with it), the player **does not acquire** the power up.

- **Game End:**
  - Upon game end, the game scene is replaced with 'Game End'/'Game Lose' based on the game state along with the game score.
  - A game lose should appear if all the lives are lost.
  - A game end appears when the time is up.
- **Animations:**
  - **Player:**
    - The player should jump **and** duck using the keyboard keys.
  - **Power-ups/Collectables:**
    - Power-ups/Collectables should animate in position(any sort of animation).
    - Both should have **different** types of animation.
    - Move towards the player position with increasing speed based on time.
  - **Background:**
    - There should be any type of animation in the background other than the health, score and time updating throughout the game
  - **Obstacles:**
    - Move towards the player position with increasing speed based on time.
- **Bonus:**
  - **Sound:**
    - Background music that starts playing when the game starts,
    - A sound effect plays when collecting objects from the scene,
    - A **different** sound effect plays when colliding with obstacles in the scene,
    - A **different** sound effect upon game win,
    - A **different** sound effect upon game loss.
  - **Textures:**
    - Apply textures to everything in the scene except for very small objects where texture images will be invisible.
  - **Note:**
    - The bonus is acquired by **EITHER** sound(all points) **OR** texture.

Submission:

- The assignment should be implemented in **OpenGL**
- This is an **INDIVIDUAL** assignment.
- This assignment is worth 7.5%
- **Deadline** for the assignment: **Friday 18th of October at 11:59 pm**.
- Cheating cases will lead to a **ZERO**, this includes:
  - Copying the code from the **Internet** or **github repo** or a **colleague**.
  - You are expected to understand **any** code generated by **ChatGPT**, as it is part of your assessment.
- Submission guidelines:
  - You should name your **.cpp file** in the following format PXX-55-XXXX,
  - **Do not** submit the **.sln** of your assignment, to avoid getting a **zero**.
  - Submission form: submit here.