Game Design & Programming I

CMPT 414 / MSCS 565 – Fall 2023

Game #2 – Variation on Pong – 100 points

Goals

To learn how to use basic C++ and SMFL by creating a simple, working video game.

Preparation

You must already have a Git repository named *{YourLastname}-work*. In Visual Studio, add to your existing *GameProg1* solution a new C++ *Empty Project*. This project will house the *Pong* clone that you will create as described below.

Requirements

Follow Chapters 5 and 6 of our textbook to build a *Pong* clone, deviating from the written instructions to distinguish your game from the base version. Your customizations must include two prescribed changes and four out of six discretionary changes listed below.

★ *Build Configuration* **5pts**

- Project organization and settings. Project resides in the same solution as your lab work. Program does not fail to compile or run due to improper config.

★ *Prescribed Change #1 – New Assets* **10pts**

- Find and use new graphics (for the bat and ball), new font, and new sounds.
- Your game should have a cohesive visual and aural aesthetic style.
- Be sure to use a *View* to ensure scaling of your game to different screen sizes.

★ *Prescribed Change #2 – Change of perspective* **15pts**

- Orient the game at a right angle to what the book described.
- This means that the Bat will move vertically across the entire height of the screen. Make sure the Bat can also move horizontally, but limit this to a small part (e.g., 20%) of the screen width.
- The Ball must then bounce horizontally, from one side of the screen to the other, bouncing off the top and bottom, of course.

★ *Discretionary Changes – Choose four of six options below.* **10pts each (max 40)**

a) **Drag** – the ball loses momentum as it travels, gradually slowing down.

b) **Spin** – if the ball hits the bat while the bat is in motion, apply *spin* on the ball proportional to the speed of the bat. Spin must alter the angle of the ball (and optionally it can make it curve).

c) **Rebound** – hitting the bat gives a small speed boost to the ball.

d) **Rewards** – impart some reward to the player if they keep the ball going for a increasingly long amounts of time. Reward could be points or more lives or something else of your choice.

e) **Environmental Effects** – add some kind of visual and/or audio effects to your game to make it more exciting. E.g., maybe the ball changes color when hitting the walls or bat, maybe the music changes the longer the ball keeps going, etc.

f) **Special Ability** – add an additional keyboard control that trigger a player ability of your choosing. You can decide what the ability is for your game – suggestions include *invincibility* (e.g, last 5s, usable very 60s), *slow time* (timer runs slower for 10s)

Requirements (cont'd)

★ *Code Correctness* **20pts**

- Program must not fail to compile or run due to syntax or logic errors.
- Program compiles and runs without error.

★ *Code Style* **10pts**

- You must use a simple object-oriented design with separate header and implementation files for your classes.
- Program demonstrates consistent, readable coding style and follows best practices for C++ including, but is not limited to, indentation and spacing; naming of variables, functions, and classes; architecture and organization; use of standard library feature.

Advice

You should break up your work over multiple days and commits. Your commit history should show your process, not just the final product. Push any local work to your remote GitHub repository regularly. Don't forget to write short but meaningful messages for every commit. (Tip: Consider using wording from these instructions as your commit messages.)

Test, test, test… and test again. Then test some more. When you think you've tested enough, go back and test yet again. Then get someone else to test for you while you test theirs. Etc.

Submitting

You must push your changes to GitHub before the due date.

Note: Pushing regularly will reduce the risk of losing your work, so do not wait until after you have made all changes and commits before pushing.