Game Design & Programming I

CMPT 414 – Fall 2023

Game #1 – Variation on Timber!!! – 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 *Timber* clone that you will create as described below.

Requirements

Follow Chapters 1 to 4 of our textbook to build a *Timberman* 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* **10pts**
- ◦ 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 (background, sprites) for your game.
- ◦ Find and use a new font for your game.
- ◦ Find and use new sound effects for your game.
- ◦ Your game should have a cohesive visual and aural aesthetic style.

★ *Prescribed Change #2 – Flexible video mode* **5pts**
- ◦ Adjust the video resolution to match your background image, if needed.
- ◦ Ensure your game properly fills the screen regardless of native screen resolution.

★ *Discretionary Changes – Choose four of six options below.* **5pts each (max 20)**

a) **Horizontal direction** – player moves up/down instead of left/right, and hazards go right-to-left (or vice versa).

b) **Three Positions** – player can occupy one of three different "positions", such as left/center/right, or up/middle/down, or left/right/crouched.

c) **Ambient Object Direction** – change the ambient objects to move in the horizontally instead of vertically.

d) **Ambient Object Behavior** – change the way ambient objects enter and exit the view, by changing where and how they start and end their movement.

e) **Dynamic Difficulty** – keep track of how often the player dies/fails and adjust the frequency of hazards to make the game easier if the player loses too quickly. You can decide on the specific meaning of "too quickly".

★ *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* **20pts**
- 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.