CSD1401 - Assignment 1

Purpose

This assignment will help students develop:

- Familiarity with the C programming language
- Familiarity with the Visual Studios IDE
- Familiarity with the CProcessing engine
- Familiarity with developing an application

Overview

Writing programs for users can be complex. We have to think about:

- How to process input from the Operating System
- How to load fonts and images
- How to render shapes on the screen
- How to apply texture on shapes
- How to even make what we have in our heads a reality!
- How to make all the above *fast* and *efficient*
- How to port all the above to different platforms
- How to port all the above
- and more...

But we all have to start somewhere.

CProcessing

This is a framework that will do most of the heavy lifting for us. It is essentially code written by someone else so that it's easier to write code we care about. Some of its features includes:

- Creating a window
- Handling input
- Load fonts and images
- Drawing shapes and text
- And more...

As such, all we need to think about is what we want on the screen!

Tasks

Create the Visual Studio Project

1. In the `src` folder, you should see `splashscreen.h`, `splashscreen.c`, `game.c`, `main.c` and an `Assets` folder
2. Use these files to create a working CProcessing Visual Studio Project.

View and understand what's required from the sample program

Open the `sample` folder and run `main.exe`. You should see a window pop out on your screen. Observe what is happening in the window. In this assignment, you will be implementing a similar program to this sample program given to you.

You can think of the program as having two separate features that you will need to implement:

1. The splash screen that fades the DigiPen logo in and out.
2. A circle slowly that follows the cursor.

Your final program does not need to be exactly the same as the example program given, but it must at least have those features.

Understanding what is a 'Game State'

In the CProcessing framework, there is the concept of what is known as a Game State. You can think of a Game State as a 'section' of a game. For example, most games might have the following Game States:

- Splash screen - Where you display the logo of your company, publishers and libraries that you use in your game.
- Main menu - Where you allow players to start the actual game, exit the game, or adjust their settings.
- Game - This is the actual game that players play your game for.
- Credits - This is where you show off the names of people who worked on your game!

A single Game State in CProcessing is defined with 3 types of functions. They are:

- **Init** - The function that will run *once* when the Game State changes, *before* the Update function is first called.
- **Update** - The function that will run every frame.
- **Exit** - The function that will run *once* when the Game State changes, *after* the the Update function is called.

In the handout given, we have already defined a single Game State for you in `splashscreen.c` and `splashscreen.h`. They are the 3 functions `splash_screen_init()`, `splash_screen_update()` and `splash_screen_exit()`, which will represent the **Init**, **Update** and **Exit** functions of a Game State respectively. The setting up of this Game State is already done for you in `main.c`.

Do note that since we only have one Game State, `splash_screen_init()` will be called once when we run our program, `splash_screen_update()` will run every frame, and `splash_screen_exit()` will simply be called once just before our program closes.

Planning the code structure of our features

Even though it sounds like a seemingly simple task, there is still the need to plan out what the code should look like before diving into details. For all features you implement in any software engineering project, we always keep things simple first by breaking down our feature into these steps:

1. At the start of the state, declare and initialize any variables you might need at the start. Load any resources (e.g. image, audio) you need.
2. For every frame until the state ends:
    1. Retrieve input data from user that you need (e.g. mouse position, etc)
    2. Update necessary variables (i.e. these might or might not be based on user input)
    3. Clear the background and draw the things on the screen.
3. When the state exits, unload any resources you loaded.

Since this assignment can be broken down into 2 features, we can concretely plan each feature individually. For example, when we think about DigiPen logo in and out, we can express the steps above more concretely into something like this:

1. Load the image once when the state begins. Create a variable that represents the image's transparency.
2. For every frame:
    1. Update variable that represents the image's transparency, increasing or decreasing its value depending on what your variable represents.
    2. Clear the background and render the image at a visible position with the image's transparency value.
3. When the state exits, unload the image.

Likewise, we can do the same for the circle that follows the cursor's position

1. Create variables that represents the circle's position.
2. For every frame:
    1. Retrieve the current mouse position.
    2. Update the variables that represents the circle's position such that it's a little nearer to the current mouse position.
    3. Clear the background and render the circle at the variables that represents the circle's position.

Looking for the CProcessing functions to use

The CProcessing wiki here contains documentation on what functions it has available for you to use. Below are some of the functions you might want to look up and use in order to complete this assignment:

- CP_Input_GetMouseX
- CP_Input_GetMouseY
- CP_Color_Create
- CP_Settings_Fill
- CP_Graphics_ClearBackground
- CP_Graphics_DrawCircle
- CP_Image_Load
- CP_Image_Free
- CP_Image_Draw
- CP_Vector
- CP_System_GetDt

Check your work

Check that your program has the following features:

1. A splash screen that fades the DigiPen logo in and out.
2. A circle constantly that follows the cursor.

Again, your final program does not need to be exactly the same as the example program given, but it must at least have those features.

Play around with CProcessing

You are highly encouraged to copy this project into a separate folder and play around with with the CProcessing framework when you have time. Reference `cprocessing.h` and try calling its functions and examine its effects!

Deliverables

1. Zip your completed `splashscreen.c` and `splashscreen.h` using the convention `<DigiPen student id>_1.zip`. For example: If my DigiPen student e-mail is `cheng.dingxiang@digipen.edu`, my DigiPen student id is `cheng.dingxiang`. My zipped folder will hence be named `cheng.dingxiang_1.zip`
2. Submit `<DigiPen student id>_1.zip` file to the appropriate assignment submission link in Moodle.
3. After submitting, do a sanity check by re-downloading the file that you submitted and ensure that it is indeed the file that you submitted.