CSC 205 - SPRING 2016
2D GRAPHICS AND IMAGE PROCESSING
ASSIGNMENT 2
UNIVERSITY OF VICTORIA

Due: Tuesday, February 16th, 2016 by noon.

1 Assignment Overview

The objective of this assignment is to design and implement a real-time graphics application, which will include user interaction, collision detection and animation. Your submission can integrate these elements in any fashion, including a 2d game, a visualization tool (for example, for geographical data) or a physical simulation. The distribution of marks is detailed in Section **??**. The criteria are intended to be open-ended, to allow you as much freedom as possible in the exact form of your final submission; if you are unsure about how a given item in the evaluation criteria will be graded, contact your instructor before the due date.

If you want to implement a game, classic games like Pong[1], Arkanoid[2] and Gradius[3] would be good choices. A simple platform game would also be viable. Note that if you want to develop an advanced game, you may want to start with a simple game for this assignment and use that as the basis for your project at the end of the semester.

You may implement your solution in any language that can be run (using non-proprietary software tools) in ECS 354, ECS 242 or ECS 266. A simple diagnostic program, with basic user interaction and frame drawing functionality, has been implemented in Java, C++ and Python. The Java version uses the built-in graphical functionality of the Java API to render graphics. The C++ and Python versions use a third-party, open source library called the Simple Directmedia Layer (SDL). The SDL is just one of many possible low-level graphics libraries, and was chosen for the sample code primarily because of the relative symmetry between the C++ and Python bindings. You may want to investigate alternative libraries, such as `glfw` (used by CSC 305). In particular, you should not feel obligated to use the SDL just because it is used in the provided examples. As with assignment 1, you are not required to use any of the provided code as the basis for your implementation.

If you use a language which relies on platform-specific libraries, you may find that it is easier to complete the entire assignment on a lab machine, instead of working on it on your own machine, since it can be difficult to resolve library version conflicts in some cases. The sample code for C++ and Python has been tested on the ECS 354 machines and a recent Debian-based linux distribution. Talk to your instructor if you need help compiling or running the sample code on your own machine. The compile/build/run process for the starter code will be covered in the lectures and labs.

---

1. `https://en.wikipedia.org/wiki/Pong`
2. `https://en.wikipedia.org/wiki/Arkanoid`
3. `https://en.wikipedia.org/wiki/Gradius`

2 Externally-Sourced Code

You are permitted to use appropriately licensed code from an external source, if full attribution is given (including the name of the original author, the source from which the code was obtained and an indication of the terms of the license). Note that using copyrighted material without an appropriate license (such as the GPL, LGPL, BSD License or MIT License) is not permitted. Short fragments of code found on public websites (such as StackOverflow) may be used without an explicit license (but with the usual attribution). If you embed externally sourced code in your own code files, add a citation in your code file and include a README file detailing the sources of all code used. If you use entire, unmodified source files from an external source, a README file is not necessary as long as the file in question contains complete authorship and licensing information. If you submit the work of others without proper attribution, it will be considered plagiarism. Additionally, for assignments in CSC 205, the following two caveats apply to externally sourced code.

- You will not receive any marks for the work of others. That is, if you use externally sourced code to implement the core objectives of the assignment, you will only be marked on the parts of the code that you personally wrote. Therefore, you should only use externally sourced code in a supplementary capacity (such as including a third-party hash table implementation which your code uses to store data, or a third party Gaussian elimination package to solve matrix equations). If you have any doubts about what is considered supplementary, contact your instructor.
- You may not use externally sourced code from another CSC 205 student, or share your code with another CSC 205 student (whether they intend to use it or not), without explicit permission from the instructor.

3 Evaluation

Submit all of your code electronically through the Assignments tab on conneX. Your submission should include a complete set of files needed to compile, run and demonstrate your implementation's functionality for marking. Your implementation will be marked out of 110 during an interactive demo with an instructor. You may be expected to explain some aspects of your code to the evaluator. Demos must be scheduled in advance (through an electronic system available on conneX). If you do not schedule a demo time, or if you do not attend your scheduled demo, you will receive a mark of zero.

You may receive up to 120 marks on this assignment, but the final assignment mark will be taken out of 110 (so any marks you receive beyond 110 will be treated as bonus). Note that the criteria listed below add to more than 120 marks, but the final mark will be capped at 120 (and some criteria items are mutually exclusive).

The marks are distributed among the components of the assignment as follows. The term 'playing field' is used below to refer to the active drawing area; this should not be interpreted to mean that you are required to implement a game. If you have any questions about how the specifications below would apply to your intended implementation, ask your instructor.

Marks	Feature
	Rendering and Animation
25	Basic functionality: The program contains a frame loop to redraw all objects on screen at regular intervals, and the frame rate of the program is sufficient for smooth animation. The program must demonstrate the ability to draw multiple sprites of various types (rectangles, circles, lines, etc.) at some point during its operation. It is not necessary for multiple sprites to always be visible. The program contains at least one animated sprite (such as a bouncing ball) which is animated smoothly without direct user intervention (that is, there must be at least one example of non-user-controlled animation).
8	Instead of plain shapes (circles, rectangles, etc.), graphical sprites (procedurally generated or loaded from images) are used to represent entities in the playing field. If you use image files from an external source, remember to include full attribution.
10	Sprite appearances are animated (that is, the appearance of each sprite changes over time, instead of just its position and orientation).
5	Moving objects have a visible rotational component (such as a circle with a rotating texture or a non-circular object which is visibly rotating).
	User Interaction
15	The user can interact directly with moving objects using the keyboard (for example, by controlling a sprite using the arrow keys). For full marks, both single and continuous events (that is, keys being held down) but be defined.
15	The user can interact directly with the objects in the playing field using the mouse. This can be implemented in different ways, including allowing the user to click on existing objects to interact with them or associating a sprite with the mouse cursor and having that sprite influence other objects. For full marks, both motion and button events must be defined.
	Object Interaction
25	A moving object in the playing field is affected by collisions with other (moving or stationary) objects and the bounds of the drawing canvas. For full marks, at least two different types of objects (e.g. circles and rectangles) must have defined interactions.
10	The playing field contains multiple moving objects, which are influenced by collisions with each other and stationary objects.
5	The user is given the option of adding an arbitrary number of mutually-colliding objects to the playing field.
10	Object interactions are modelled with realistic physical simulations (for example, by modelling collisions with force equations, and associating larger objects with larger mass for momentum transfer).
15	Object interactions incorporate angular momentum (requires visible rotations). Note: This is not easy.
	Immersion
20	For games: The game has well defined rules and objectives, and challenges the user. The game should either have a defined 'winning' condition or should escalate in difficulty indefinitely. The controls should also be intuitive and easy for new users to understand.
10	For games: Artificial intelligence techniques are used to implement an adversary to replace a human second player, or to adaptively adjust the environment in a single player game based on the player's behavior.
20	For simulations and visualizations: The interactive element of the program allows the user to interact naturally with the displayed environment, and the parameters of the environment (or input data) can be modified.
	Miscellaneous
15	Code efficiency: The program contains non-trivial optimizations which significantly reduce CPU or memory consumption, or significantly increase rendering performance. To receive these marks, you will be expected to demonstrate the performance of the code both with and without the optimizations.

Additionally, you may receive extra marks (with the total overall mark capped at 120) for other features of your own design, if you have received permission from the instructor before the due date.

Ensure that all code files needed to compile and run your code in an ECS lab are submitted. Only the files that you submit through conneX will be marked. The best way to make sure your submission is correct is to download it from conneX after submitting and test it. You are not permitted to revise your submission after the due date, and late submissions will not be accepted, so you should ensure that you have submitted the correct version of your code before the due date. conneX will allow you to change your submission before the due date if you notice a mistake. After submitting your assignment, conneX will automatically send you a confirmation email. If you do not receive such an email, your submission was not received. If you have problems with the submission process, send an email to the instructor **before** the due date.