**CSC171 — Project 3**

**Videogame!**

In this project you will put your your programming, graphics, and animation skills together to develop a videogame. You have two choices of game – a drone piloting game, or something you invent yourself. This document will describe the general requirements, but many of the details are going to be up to you. Take the basic ideas and run with them to develop something interesting. Creativity counts. So does good software design and clean, well-documented code. Include a writeup with instructions and a description of what you did (README or PDF allowed this time – still no docx, etc. The PDF is because you might want to include screenshots.) **Y**ou may work in a group of up to four students on this project.

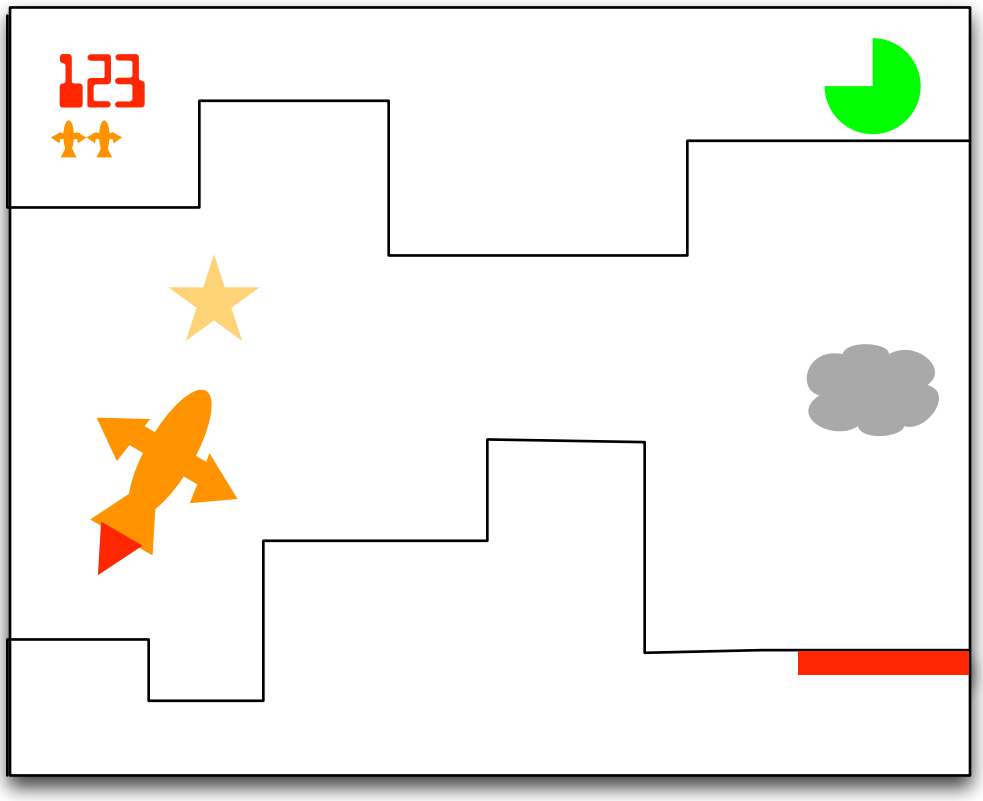
# Duedate

1. November 25th 1159pm for +5 points,
2. December 2nd 1159pm for +3 points,
3. December 8th 1159pm HARD DEADLINE (submit what you have for partial credit).

# Drone Pilot

**Drone Pilot** – A game where the goal is to pilot a drone vehicle through a cluttered environment.

In Drone Pilot, the player has to steer an aerial drone through a cave and land safely in a landing zone. The drone has two small rockets, one on each side, that can make it rotate, and a larger rocket that propels it forward (in the direction it is facing). The cave has jagged rocks sticking up from the ground or down from the ceiling of the cave. A rough mockup of what this might look like is shown below.




The key to getting the motion of the drone right is to use the equations of motion similar to those for a projectile. The side rockets only change the angle at which the main rocket fires. The main rocket changes the acceleration according to the drone's angle of rotation (horizontal component is $a\cos(\theta)$, vertical component is $a\sin(\theta)$). The acceleration changes the velocity and hence the position in time. There is also gravity, so the drone will fall and crash unless the main rocket is fired. If the drone lands while going too fast it also crashes.

I suggest using the arrow keys (or another set of four keys) to control the drone. The left and right arrows each fire one of the rotation engines (so left and right arrow rotate the drone). The up arrow fires the large engine, making it go up when not rotated, otherwise in the direction it is pointing. You may want some additional UI (toolbar, menus, buttons, etc.) to control the game.

The player gets some number of "lives" (drones) to start (three would be good). Your game should show the player's score and the number of lives/drones remaining, and update these appropriately as the game is played.

Finally, the drone has a fuel tank that empties as the rockets are fired (this is the green circle shape in the mockup). The player must land safely before running out of fuel. If so, they "win the level" and receive bonus points. A new level is created and displayed and they get a fresh supply of fuel to complete the mission. Presumably successive levels are harder in some way, such as having less fuel, a more challenging course, etc.

For this game, start with one or a few simple levels whose "rocks" are designed and not random. Using rectangular "rocks" (as in my mockup) makes detecting when the drone hits them easier. Maybe don't have rocks on the ceiling to start. With some creative drawing or images, you could probably make even rectangular rocks look interesting. You will have to decide how to represent a level (likely with a collection, or a file, or a custom class.) Other shapes would make for a better game though...

# Flourishes

We would prefer to experience some variety when grading the projects, and would like to give you an opportunity to choose your own path. Each submission must also implement one flourish option from the list below:

- Randomized stationary obstacles that destroy the drone. These should have some randomness, so that different games and different levels within a game look different. (This one is easiest).
- Non-stationary objects, like clouds, spaceships, or twirling stars, that score extra points (and vanish) if hit by the drone.
- Two player mode: two drones, controlled with different sets of keys. For Drone Pilot, you need to do something about the drones bumping into each other (I would suggest looking up elastic collisions to make them "bounce" off eachother. This may be tricky.)
- Sound effects: Plenty of things you can do using the `javax.sound` classes. It can be a bit tricky to do this without blocking the UI or the animation timer. You may find it useful to learn about `Threads`.
- High scores list: You need the UI for entering a new high score and viewing the high scores, and of course it has to persist between runs of the game. (This one should be easy too).
- Make it look cool: you can learn about polygon filling and textures to make your game actually look good. (I.e., nifty backgrounds and objects that don't just look like squares and rectangles.) (This one can range from easy to high effort.)
- Other ideas: have something that seems genuinely interesting and non-trival? Document what you did *clearly and prominently* in your documentation. We'll give you credit based on our assessment of how hard it was to do. The professor reserves all rights to reject suggestions, but you can email or stop by office hours to discuss your ideas.

## Something You Invent

You may also invent your own game as long as it meets the following requirements:

- There is animation.
- It is interactive (i.e., keyboard/mouse.)
- There is a scoring mechanism and it is shown to the player.
- There is a definitive ending mechanism (i.e., you can win or lose).
- There is a physical mechanism (i.e., velocity, mass, motion, gravity).
- There is collision detection (i.e., walls, items, rewards, hazards.)
- It is creative.
- You implement a flourish (randomized collidables, two-player mode, sound effects, high scores list, it looks really good, or some other tricky thing.)

# Academic Honesty

Searching online for solutions (e.g., via github) is STRICTLY FORBIDDEN and would be considered as a significant instance of academic dishonesty. Any submission of plagiarized code (even with modifications) will be reported to the board of academic honesty and penalties will be applied. For this project, it is unlikely you can find solutions online; however, you should still not attempt to seek them out.

You ARE ALLOWED to research specific subtopics – e.g., collision detection, sound, random number generation, threading, etc. You are NOT ALLOWED to go find a similar game on github and submit it (or a modification) as your work. Any penalties for group submissions with plagiarized work will be applied to all members of the group. It is your responsibility to act ethically.

# Submission Instructions

Only ONE member of a group should submit the work. The work MUST be a zipfile with a README that begins with the names and netids (email) of all members of the group, otherwise we won't be able to give you grades.

The zipfile must also include the Java source files of all the classes you built to complete your project. You may choose the name of the main class but you must describe how to run your program in the README. The README should also describe the classes you wrote and their purpose/what they do.

Your README must describe the flourish you chose to implement, and how you did it. If your game is not drone pilot, you MUST also describe your game and game mechanics in the README. In particular, you MUST outline POINT-BY-POINT how it meets the requirements of the above section. (If you are doing Drone Pilot, including this same list will help us grade your submission. This is particularly important if you aren't able to finish and are looking for partial credit.)

We are hoping to grade these quickly and that almost all grades will be nearly 100%, so help us help you by completing all requirements and communicating exactly how you completed them!

Please reread this section a couple times before submitting your work.