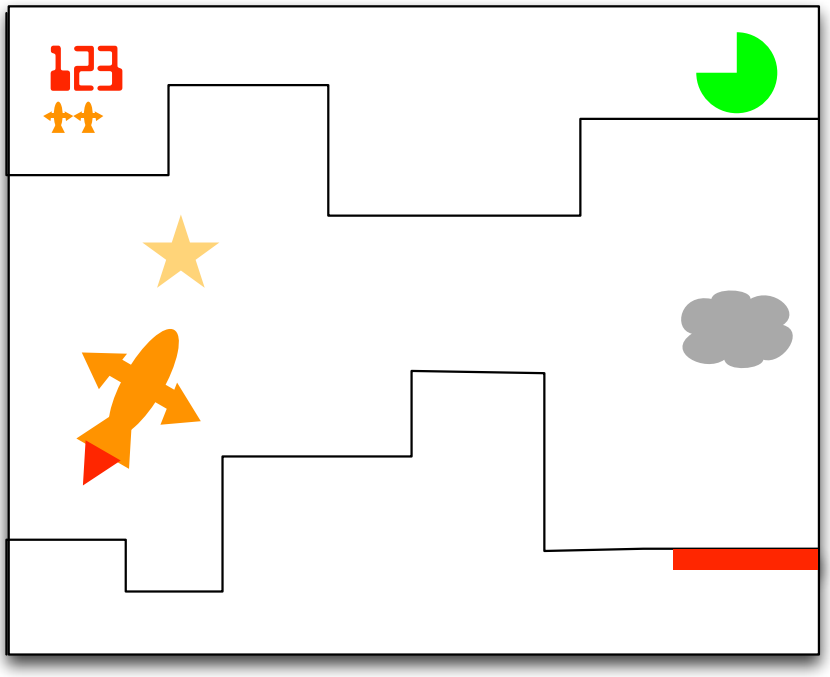
CSC171 — Videogame Project



In this project you will put your your programming, graphics, and animation skills together to develop a videogame. You are encouraged to work in teams of up to four students for this project and to treat it as an opportunity to collaboratively build something you are proud of. All teams are required to demonstrate their games during lab; therefore teams are encouraged to be registered in one lab. In order to help you make adequate progress, you will be asked to complete an online check-in and an in-person TA check-in to ensure you're on a good track. In this document we suggest several possible projects you might consider, and offer a ready-made example which you may choose instead: Space Pilot.

Objective

Your objective is to design and implement a playable game. Your game must have a minimum threshold of complexity, but otherwise you are encouraged to think about games you may have enjoyed playing as a kid (or even now) and consider programs inspired by those games. Even though you will be building a game to play on a computer, you do not have to limit it to traditional "video games". Suggested game styles include:

1. platformer – create a sprite, navigate a side-scrolling world, acquire powerups, avoid obstacles/danger, reach a goal. Note: no snake games.

2. cards/dice – create a competitive card game, design your own cards and rules or pick an existing card or dice game

3. board game – create a grid based world and a set of arbitrary rules for progression (battleship, monopoly, checkers, chess, go). Note: no tic-tac-toe games.

4. educational game – create a math game for practicing arithmetic, vary the speed and challenges, make it interactive; create a memorization game (simon says).

5. other game – suggest something else to the instructor. Quality/creativity needs to be higher for this option.

In addition to the game itself, all teams must submit two written documents along with their programs. You are encouraged to include diagrams and screenshots, so PDF format is recommended. (Please do not submit docx). The required documents are:

- a user manual, which explains how to play the game.
- a design document, which describes each of the implementation classes and their relationships.

Finally, your entire team must work together to present your work in lab. Presentations should include a live demonstration of playing the game, and a brief discussion of highlights from the design document, as well as a brief opportunity for questions from fellow students. (Please be sure to attend the presentations of your fellow students in lab, and please be respectful and kind as an audience member during all presentations!)

Sample Game: Space Pilot

Space Pilot – A game where the goal is to pilot a space vehicle through a cluttered environment.

In Space Pilot, the player has to steer a spaceship through a cave (you know, in space) and land safely in a landing zone. The ship has two small thrusters, 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 at the top of the page.

The key to getting the motion of the ship right is to use the equations of motion similar to those for a projectile. The side thrusters only change the angle at which the main rocket fires. The main rocket changes the acceleration according to the ship'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 ship will fall and crash unless the main rocket is fired. If the ship lands while going too fast it also crashes.

I suggest using the arrow keys (or another set of four keys) to control the ship. The left and right arrows each fire one of the rotation engines (so left and right arrow rotate the ship). 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 "rebuilds" (ships) to start (three would be good). Your game should show the player's score and the number of rebuilds/ships remaining, and update these appropriately as the game is played.

Finally, the ship has a fuel tank that empties as the thrusters 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 the mockup) makes detecting when the ship 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

Each submission must also implement **one flourish per team member** option from the list below:

- Randomized stationary obstacles that destroy the ship (or have some other negative effect). These should have some randomness, so that different games and different levels within a game look different. This is a good opportunity to use drawImage for sprites. (This one is easiest).
- Non-stationary objects, like clouds or twirling stars, that score extra points (and vanish) if hit by the ship. Be sure to tune the fps/velocities/responses so that it feels like a game - a little challenging, but not impossible. This is also a good opportunity for sprites.
- Two player mode: two ships, controlled with different sets of keys. For Ship Pilot, you need to do something about the ships bumping into each other (Consider 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-trivial? Document what you did *clearly and prominently* in your documentation. The instructor reserves the right to disagree with something being non-trivial, so you are encouraged to discuss with them or with TAs early on for feedback.

General Platformer Requirements

Space Pilot is one example of a platformer. You are encouraged to design your own, or implement something inspired by one of your favorite games. In order to meet the threshold complexity requirement, consider the following properties:

- There should be animation/smooth motion.
- It should be interactive with keyboard (at least a few keys) and/or mouse (clicks, drags, mousewheel, etc). More than one "kind" of input or action is required.
- There is a scoring mechanism and it is shown to the player – so it's clear if you're winning or losing. I.e., you have objectives.
- There is a definitive ending mechanism (i.e., you can win or lose, or "best in 30 seconds", etc).
- It has "physics"(i.e., a spaceship with velocity and mass, gravity pulling down after a character jumps, throwing a ball).
- There is collision detection (i.e., walls, items, rewards, hazards.)
- It is creative or looks cool or is fun.
- You implement one flourish per team member (randomized collidables, two-player mode, sound effects, high scores list, it looks really good, or some other tricky thing.)

Presentations and Teams

As a special requirement of this project, you must give a *public presentation* of your work during your lab session. All team members must participate in the presentation. You will demonstrate your game to your classmates and describe your design and development process – in particular, you should be prepared to discuss your class and object hierarchy, and how you implemented the cool things in your game.

Because the presentations will occur during lab, you are encouraged to work with students from the same lab. You can form groups from other labs, but you all MUST be present for the presentation, which will be in the section with majority members. Otherwise, the presenting section must be decided by the group in consultation with TAs. This is a large class, so your cooperation in team formation and presentation scheduling will be greatly appreciated.

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.

AI

You are ALLOWED to use AI coding tools (ChatGPT, Copilot, ...) to enhance your creativity for this assignment; however, all of your final code submitted must be authored by your group. All members of a team are responsible for understanding, and being able to explain, the final project submitted by the group. If you use AI tools you are requested to document your use of AI in your design manual - please clearly identify which aspects of your work were generated or informed by AI tools.

Grading

All submissions meeting the requirements will receive full credit for implementation. Programs which are buggy or incomplete will receive partial credit. Because of the open-ended nature of this assignment, it is not possible to provide a more detailed breakdown of scores in advance.

Your overall grade on the project will be composed as follows:

- 5% Proposals (due Oct 30th)
- 10% TA Check-Ins (by Nov 22nd)
- 65% Implementation (due Dec 2nd)
- 20% Presentations (during Dec 2nd - Dec 5th)

Submission Instructions

We're going to try gradescope's "group submission" option. Exact submission instructions TBD, but submission deadline will be Dec 2nd 1159pm.