Course Missive

Fall 2021

Introduction

Welcome to CSCI 1950n, 2D Game Engines. This document provides you with a lot of important information about the course, so it is essential for you to read and understand it. You may also want to refer to the course syllabus, the calendar, and the list of assignments, all of which are available at the course website (cs.brown.edu/courses/cs1950n/).

CS1950n is a student-run course. The TAs handle course design, lecture, office hours, and evaluation duties, working with the advising professor, James Tompkin, as needed. He is also available for student administrative situations like Health Services and Dean's notes, SEAS requests, evaluation disputes, and final grade posting.

Because students cannot sign up for the course without an override, the TA staff will maintain the class list (and wait list if needed). In order to ensure that every student has a positive experience in this course and gets the attention and assistance they need from the course staff, enrollment has been capped at 40 students. If enrollment reaches this limit and additional students still wish to take the course, the staff will decide which students will be admitted, with preference given to seniors.

Slack

This year, we will be using Slack for announcements, questions, and other communications throughout the semester. The link to join will be distributed on the first day of class.

Course Goals

In this course you will learn techniques needed to create 2D game engines, including animation, simple AI, collision detection, physics, and raycasting. You will create a 2D game engine over the course of the semester, adding a few features to it each week. At the same time, you will also create a series of games using your engine that demonstrate the use of the features you add. Near the end of the semester, you will design and

implement a final project that uses your game engine to create a finished, entertaining game.

Course Structure

Lecture will be held Mondays 3-5:30 in CIT 506. We do not anticipate using the full time block for lectures. In addition to the lectures, each student is expected to attend a 15-minute design check for each assignment with one of the teaching assistants to explain how they plan on solving the major concepts of the week at a high level. Out-of-class work is estimated at around 15 hours per week, including the final project.

Course Staff

Professor
James Tompkin (CIT 547, james_tompkin@brown.edu)

HTA (cs1950nheadtas@lists.brown.edu
Alexander Ivanov (aivanov6)

UTAs (cs1950ntas@lists.brown.edu)
Carlos Perez-Ruiz (cperezru)
Sal Brandi (sbrandi)

Prerequisites

The official prerequisite for this course is a completed CS intro sequence (CS 15/16, CS 17/18, or CS 19), or have Professor Tompkin's permission. CS 32 is not a requirement, but having experience with large scale projects will make this class easier.

The most important skills for you to have in order to do well in this course are:

- **Comfort with Java**. You should be able to design, program, and debug efficiently in Java, and read and understand Java documentation.

- **Object-oriented design**. You should be comfortable designing a system of cooperating objects to represent an abstraction or solve a problem.

In addition, experience with the following will be helpful and will make your life easier in this class:

- **Large-scale projects**. It is strongly recommended to have some experience designing, implementing, and debugging non-trivial (over 2000-line) code systems, such as the multi-person projects in CS 32. Although CS 32 is not a prerequisite, having completed it will make this class much easier. If you haven't taken CS 32 you will be learning these skills in addition to course material.

- **Vector arithmetic**. This class depends heavily on high school-level vector math (adding, subtracting, dot products, normalizing, etc.), so it helps to have a working knowledge. If you are not comfortable with vectors, though, don't let this prevent you from taking the class – the TAs can explain everything you need to know about vectors.

- **Physics**. Some knowledge of high school-level physics, such as momentum, velocity, and forces, will help with the projects involving physics simulation.

- **Git and GitHub**. GitHub Classroom will be used to pull stencil code, submit assignments, and receive grades. We'll give you all you need to get this part set up, but familiarity with Git will be helpful. If you choose to do the final project in groups, more experience with version control systems will also help synchronize your code between group members.

- **JavaFX.** It is not required to have a good understanding of JavaFX's UI toolkit, even though this class involves creating graphical applications in Java. You will be provided with support code that abstracts away most of the complications of using JavaFX to draw a window on the screen.

Projects and Grading

The course consists of five programming projects: four regular projects, and a group final project. Each project is divided into one or more weekly checkpoints. There are no homeworks or exams. Assignments are assigned on Mondays after the lecture and are due the following Monday at 11:59 pm. Assignment checkpoints are the Tuesday and Wednesday after the project is assigned. Assignments should be submitted via GitHub Classroom.

Here is a brief overview of each project:

Project	Type of Game	Content You Will Implement
Tic	Tic Tac Toe	Architecture, Graphics
Alc	Alchemy	Components, Sprites
Wiz	Top Down / Pokemon-like	Collisions, AI, World Generation
Nin	Platformer	Animation, Physics
Final	Whatever you want ;)	Whatever you want ;)

Each project (except for Tic) is split up into weekly assignments (ex. Wiz I and Wiz II), to pace and spread out the requirements of each project. Each checkpoint has three sets of requirements: design check (worth 1 point), global & primary requirements (worth 2 points), and secondary requirements (worth 1 point). The design check, primary, and secondary requirements make up the four points that each assignment is worth. You cannot receive credit for secondary requirements unless you have met the primary requirements.

Each student gets three “retries” for the semester. Note that you will not/cannot lose credit by using a retry. There are two ways to use a retry:

- You may use a retry to hand in a project up to a week after its due date.
    - To do this, simply turn in the assignment late.
- You may use a retry to re-hand in a project up to a week after you receive your grade report. You must have attended your design check to use a retry this way.

Note, however, that these projects are cumulative. The engine features that you implement in one week will generally also be necessary in order to complete the next week’s project. Since each project depends on the previous one, you will still need to complete them in order, and once you are late on one project you run the risk of staying behind schedule on every subsequent project. Do not let this happen!

This grading system can be a little confusing at first, so don’t hesitate to email the TAs if you have any questions.

All checkpoints have the same weight. Your letter grade in the course is determined by the number of points you have by the end of the semester. There are a total of 43 points you can earn.

Assignment	Points	Assignment	Points
Tic	4	Nin II	4
Alc I	4	Final I	3
Alc II	4	Final II	4
Wiz I	4	Final III	4
Wiz II	4	Final IV	4
Nin I	4		

Points Earned	Points Missing	Grade
35 - 43	0 - 8	A
27 - 34	9 - 16	B
20 - 26	17 - 23	C
< 20	> 23	NC

Regardless of the number of points you have, you must have an engine that satisfies all primary requirements from every checkpoint by the end of the semester in order to receive a passing grade.

Incompletes (grades of INC) will not be given except in extenuating circumstances authorized by a dean or a note from Health Services. If you know in advance that you will be requesting an incomplete grade, please talk to the professor as soon as possible. There is no grading curve for the distribution of final grades. All final grades will be determined using the table above. All projects are graded by TAs.

If you think there is a mistake in the grading of an assignment, you should first talk to the TA who graded that assignment. If you are still unhappy, you may contact the HTA. If the HTA is unable to resolve the problem, please contact Professor Tompkin.

Final Project

The last project in this course will be an open-ended project, for which you are encouraged, but not required, to work in a group. Unlike the other programming projects, you will determine the requirements for this assignment. You will be able to pick from a list of engine features to implement and then write your own game requirements based on the kind of game you want to create. While some of the other projects will focus more on implementing engine features than gameplay, the goal of the final project is to use your engine to create a fun and exciting game. It's important to put some thought into what kind of game you want to create and how you will make it enjoyable to play, so start thinking about the final project as early as possible.

An idea for your project will be due at the same time as the last weekly checkpoint for Wiz, and a more detailed design proposal, including your list of requirements, will be due at the same time as the first handin of Nin.

If you want to work in a group, this will also be the time to form one; some class time will be dedicated to helping people finalize project groups the week before the design proposal is due. Due to the remote nature of this course, it will be possible to do the final project on your own, however, everyone is strongly encouraged to form at least a two-person group so that you will have the resources to make a more interesting game.

Design Checks

There will be a mandatory design check for every assignment, held on the Monday, Tuesday, and Wednesday of the week the assignment is released. Each design check will last 15 minutes; refer to our website about how to sign up for one. Design checks will be worth a part of your grade, but will not be scored on correctness; we just want to see that you've thought about how to approach this project. The design checks in this course are informal.

It is important that you have thought about the project, understand the concepts that it depends on, and have a plan for solving the major problems that it involves. Questions for each design check will be posted with the assignments. The TAs will expect you to answer these questions at a conceptual level. While it is not necessary to have prepared a design diagram or have any code written, feel free to bring either to your design checks to help answer the questions.

Demos

Traditionally, this class has had an additional playtesting component where students play each other's games and give feedback. It is meant to be a fun way to find out what kind of games your peers have created, and gauge how well your project works and find non-obvious bugs before they become a problem.

This year, in person playtesting for assignments will be replaced by recording demos of your game and uploading them onto Slack (however we hope to have some kind of event for the final projects). You should provide some sort of commentary pointing out more unique aspects of your engine or game, either as an audio narration, text overlay, or written paragraphs sent alongside the demo. These videos can be informal; phone recordings are perfectly acceptable.

These demos should only feature your project, not any code you've written!

Collaboration Policy

In order to make sure that each student is graded as fairly and individually as possible, the course staff have written a collaboration policy by which we expect all students to abide. Please read this policy carefully, as it may differ from collaboration policies in other CS classes you have taken. The policy isn't too long, and we have tried to make it easy to read.

CS 1950n involves some challenging software design problems for which there is often no single "right" solution. Thus, it is helpful and encouraged to discuss the projects with your peers and help each other find more creative solutions to these problems. For these reasons, the collaboration policy is generally very lax—you are allowed to review course material with classmates, weigh the costs and benefits of competing implementations, and discuss high-level bugs you may be having. However, it is important that the work you hand in is your own. Sharing code with other students is strictly prohibited, except in the specific case of giving or receiving debugging help from another student. When you are debugging someone else's code you should not have your own open, and should only interact with the code via screen share. You should **never** copy code that's not your own. There is a lot of freedom when designing and writing your engines and games; we expect to see your unique coding style throughout.

TA Hours

TAs will hold weekly hours for questions and design checks. You can find the schedule on the course website. You can go to TA hours to ask questions about the concepts and algorithms presented in class, get advice on the design of your engine, and ask for help in solving particularly difficult bugs. TAs are here to help you, but remember, TAs are students too. Please don't ask them questions outside of official TA hours. This includes asking them in person or electronically while they are at home or in the lab. If you need to contact the TAs outside of TA hours, you can email the alias cs1950ntas@lists.brown.edu or use slack. You should generally use this alias instead of sending email to TAs individually, as most questions can be answered by any TA and you are more likely to get a timely response by emailing the alias. Similarly, if you are messaging TAs on slack message all of the TAs together. If TA hours are rescheduled or canceled for any reason, there will be an announcement to the class email list. If you feel you can't possibly make the scheduled TA hours (especially after a reschedule), get in touch with the Head TA.

Inclusivity Statement

Creating an inclusive educational environment that embraces diversity is a matter of utmost importance. We want to ensure that all students feel welcome and capable of excellency in this course. The TAs have undergone training in diversity and inclusion; all members of the CS community, including faculty and staff, are expected to treat one another in a professional manner. If you feel you have not been treated in a professional manner by any of the course staff, please contact the TAs, Professor Tompkin, Professor Cetintemel (the department chair), or Laura Dobler (the departments coordinator for diversity and inclusion initiatives). We take all complaints about unprofessional behavior seriously.

Accommodations

If you have a physical, psychological, or learning disability that could affect your performance in the course, we urge you to contact SEAS (brown.edu/campus-life/support/accessibility-services). We will do whatever we can to support accommodations recommended by SEAS. Accommodations are not typically retroactive, so students should seek help as early in the semester as possible.