Course Missive

Fall 2018

Introduction

Welcome to CSCI 1971, Topics in 2D Game Engines. This document provides you with a lot of important information about CS1971, 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/cs1971/`). Announcements made during the semester will be sent to the class email list, as well as posted on the home page of the course website.

CS1971 is a student-run independent study course. The TAs handle course design, lecture, office hours, and evaluation duties, working with the advising professor as needed. She is also available for student administrative situations like Health Services and Dean's notes, SEAS requests, evaluation disputes, and final grade posting. According to Brown Faculty Rules, a faculty member can have no more than six students who are pursuing a similar independent study; therefore, toward the end of shopping period, students will be placed in a section with a CS faculty member who may not be directly involved with the course. Because students cannot sign up for the course without an override, the TA staff will maintain the class list (and wait list if needed) until students can be assigned to faculty sections.

In order to ensure that every student has a positive experience in this course and gets the attention and assistance he or she needs 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.

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

Lectures are held once a week for a period of 2.5 hours. The last 30 to 60 minutes of the lecture are used for playtesting, a process in which all students will get to play and provide feedback on other students games. In addition to the lectures, each student is expected to attend a 15-minute design check each week with one of the teaching assistants to explain how he or she plans on solving the major concepts of the week at a high conceptual level. Out-of-class work is estimated at around 15 hours per week, including the final project.

Course Staff

Professor	Office	Email
James Tompkin	CIT 547	`james_tompkin@brown.edu`

Head TA	Login
Cass Zegura	`czegura`

Grad TA	Login
David Mayans	`dmayans`

UTA	Login
Ben Gabinet	`bgabinet`

Prerequisites

The official prerequisite for this course is that you have completed a CS intro sequence (CS 15/16, CS 17/18, or CS 19), or have Professor Tompkin's permission. 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.
- **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.
- **Version control systems.** As many people will do the final project in a group, experience with version control systems will help synchronize your code between group members.

It is not required to have a good understanding of Java's Swing 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 Swing to draw a window on the screen.

Class Meetings

Lectures are on Mondays from 3-5:20pm in CIT 506. If the time or location changes, we will make an announcement at the preceding class and send a message to the class e-mail list.

Approximately the first 30-90 minutes of this time block is used for a lecture on that week's game engine topics. After that, class will move down to the Sunlab, where the remainder of the class period is used to playtest the assignment that is due that week. Playtesting is required in order for your project to be considered complete on time, so class attendance is effectively mandatory.

Projects and Grading

The course consists of five programming projects: four regular projects, and a student-directed final project. Each project is divided into one to five weekly checkpoints. There are no homeworks or exams. Projects/checkpoints are assigned each Tuesday after the lecture and are due the following Monday night (at 11:59 pm). Assignments may be submitted by running the `cs1971_handin` script from the project's root directory on a CS department computer.

There are three sets of project-specific requirements for each checkpoint – playtesting, primary, and secondary requirements. Playtesting and secondary requirements are worth one point each, and primary requirements are worth two points. Thus, each checkpoint offers a total of four points.

Before the final project begins, there are three mini-checkpoints (to hand in your idea, your group, and your design). Each mini-checkpoint is worth one point.

If a handin does not receive points for the primary or secondary requirements, you have one week to retry the assignment and submit another handin that meets the checkpoint's requirements, **but only if you attended your design check!** If and when you retry, the handin will be regraded. If it meets the primary requirements, you will receive full credit for them. If it meets the secondary requirements, you will receive full credit for them. You cannot lose credit for using a retry. If you hand in a project late, you can still receive full credit for the primary and secondary requirements. Playtesting requirements and mini-checkpoints cannot be retried.

Furthermore, you have two extra retries, each of which allows you to retry a previous retry. This means if your retry handin still does not satisfy the primary or secondary requirements, you may use an extra retry to fix it and submit it again for grading. You may also hand in your project two weeks late and state that you want to use your standard and extra retry at the same time.

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 semeseter.

Points	Missing	Grade
42-55	0-8	A
33-41	9-17	B
25-32	18-25	C
24-	26+	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.

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 HTA 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 have a problem with the grade you received for 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 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 Tac, and a more detailed design proposal, including your list of requirements, will be due at the same time as the second 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. Although it is possible to do the final project on your own, 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 checkpoint, held on the Wednesdays and Thursdays preceding its due date in CIT 203. Each design check will last 15 minutes. You will sign up for a design check using `cs1971_signup <project>` in the terminal.

Design checks do not represent a percentage of your project grade. Instead, they must be completed to receive a standard retry for the checkpoint. If you do not complete the design check for a specific checkpoint, you will have to use an extra retry to retry that checkpoint.

The design checks in this course are informal. To receive credit for a design check, you need to answer a few questions regarding the assignment's content. It 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.

Playtesting

Playtesting is an important part of this class. In addition to creating a game engine, you are also creating playable games, and you will want to make sure that other people can play them successfully. Even in the checkpoint weeks when you are not required to have a finished game, your partially-completed game should still be a functional demonstration. For the week the final version of a project is due, playtesting should be a fun way to find out what kind of games your peers have created, and receive valuable feedback on the game you created.

Since projects don't receive numeric grades, playtesting is an important opportunity to gauge how well your project works and find non-obvious bugs before they become a problem. During playtesting, you are encouraged to try to break the games you play and report any bugs or odd behavior.

As mentioned previously, playtesting will occur during class time after lecture. Students will be randomly split into roughly equal-sized groups, and each student will playtest every other student's game in the group. You will need to fill out a playtesting form for each game you playtest, so that the person you're playtesting has a written record of your feedback.

Collaboration Policy

In order to make sure that each student in cs1971 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.

Cs1971 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 this reasons, the collaboration policy is generally very lax- you are allowed to review course material with classmates, weight 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 or copying code from other students is strictly prohibited. There is a lot of freedom when designing and writing your engines and games; we expect to see your unique coding style throughout.

As with other CS classes that prohibit sharing code between students, you are responsible for ensuring that the permissions on your source code directories do not allow other students to view them. Ask a consultant or a TA for help with permissions if you are unsure of how to use them.

TA Hours

TA hours will be held Fridays, Saturdays and Sundays in CIT 203. Once the TAs work out their own class schedules, the exact hours will be posted on the course web page. 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 talking to 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, e-mail the alias `cs1971tas@lists.brown.edu`. You should generally use this alias instead of sending e-mail to TAs individually, as most questions can be answered by any TA and you are more likely to get a timely response by e-mailing the alias.

If TA hours are rescheduled or canceled for any reason, there will be an announcement to the class e-mail 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.

Special 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.

Last edited November 11, 2018