COURSE SYLLABUS

Course Number:	CS179.14		
Title:	PC (and Console) Game Development		
Department/Program:	DISCS	**School:**	School of Science and Engineering
Semester:	2nd	**School Year:**	2014-2015
Instructor/s:	Wilhansen Joseph B. Li <wil+cs179.14@byimplication.com>		

A COURSE DESCRIPTION

The course focuses on the fundamentals of PC game programming to complement the skills learned mostly in CS177: Computer Graphics Programming but may also use some concepts in CS162 and CS130. Students will learn how to build a game from scratch in order to gain a deep understanding of their architecture and components, as opposed to using a pre-made game-making software. The format of the lesson will be a mix of lectures followed-by hands-on implementation. By the end of the semester, the students should be able to produce a non-trivial game.

B COURSE OBJECTIVES

At the end of the course, students should:

- be familiar with creating, compiling, and debugging native applications on various platforms.
- be able to prototype gameplay in a short amount of time.
- know various architectures and apply them to make their game scale.
- apply mathematics to solve certain challenges in game programming.

C COURSE OUTLINE AND TIME FRAME

The instructor reserve the right to make adjustments to the course outline as he deems fit.

Week and Topic	Learning Objectives	Activities	Student Output
1: Introduction and Git Crash Course • Game categories, components and architecture. • Git • Git workflow	• Know the different types of games and the possible divergence in their implementations. • Identify game components from a software engineering perspective. • Know different game platforms. • Know how to use git to submit homework	• Lecture • Git hands-on	Recitation and Homework
1.5: C/C++ Review • Pointers, Memory management, Constructors and Destructors	• Know the memory model of C/C++. • Be able to track down constructor and destructor execution and use them for the RAII pattern.	• Lecture	Recitation and Quiz
2: Windows Programming and Native Compilation • Compiler flags • Windows Message Pump • Static and Dynamic Libraries • OpenGL or DirectX initialization.	• Compile a native Win32 Program • Know the anatomy of a native Windows Program • Link to static and dynamic libraries • Setup DirectX or OpenGL	• Lecture • Progressive hands-on exercises	Recitation and Quiz

3: Game Loop and Timing • System clocks • High-resolution clocks • Fixed time step and flexible time step loops	• Know the different types of system clocks • Query for high-resolution monotonic clocks • Create a fixed-time step loop • Create a free-running game loop	• Lecture • Hands-on exercise	Hands-on exercise output
4-5: Physics and Collision Detection • Vectors • Circles • Axis-aligned and oriented bounding boxes • Polygons • Separating axis theorem • Kinematics and dynamics	• Use the separating axis theorem to detect collisions • Build simple simulations using dynamics and kinematics • Perform appropriate collision responses	• Lecture • Hands-on exercise	Physics simulation
6: Spatial Data Structures • Uniform grids • Quad trees	• Create a uniform grid and quad tree • Use the data structures to cull items in querying	• Lecture • Hands-on exercise	Physics simulation
7-8: Input and Character Control • Keyboard and mouse input • Gamepad input • Character control/physics	• Query for keyboard and mouse input • Query for gamepad support and input • Create a basic 2D platformer	• Lecture • Hands-on	2D platformer
9: Interpolation • Interpolation definition • Function manipulations • Continuity and differentiability • Bézier Curves	• Manipulate functions to fit interpolation requirements • Use interpolations to smoothen animations. • Fit more complex data data	• Lecture • Hands-on	Interpolation library
10: Entity Framework • Entities • Component-based architecture • Entity interactions	• Create an entity framework • Convert an entity framework to make it component-based • Accomodate entity interactions	• Lecture • Hands-on	Entity framework
11: Event Systems • Observer pattern • Generic Observer	• Create an event system • Integrate the event system into the entity framework	• Lecture • Hands-on exercise	Event system
12: Game State Framework • Game states • State pattern • Game state switching • State interpolation	• Create a game state framework • Make the game state framework support inter-state animations	• Lecture • Hands-on exercise	Game state framework

13: Configuration Management • Windows Registry • Windows user and application directories • Configuration files (INI, XML, JSON) • String parsing	• Read and write from the registry. • Know the proper directories to store configuration • Generate and parse configuration files	• Lecture • Hands-on	Configuration framework
14: Data-Driven Design • Image loading • Spritesheets • Descriptor files • File parsing	• Externalize level data to files • Describe entities and levels using external files	• Lecture • Hands-on	Hands-on output
15: Audio†			
16-17: GUI and HUD†			
18: Finals	Application of lessons	Project defense	Project

†Will be taken when there is enough time.

D REQUIRED READING

Any one of the following will suffice:

- Game Engine Architecture, by Jason Gregory, Jeff Lander and Matt Whiting, A K Peters (ISBN-13: 978-1568814131), 2009
- Computer Graphics using OpenGL, 2nd Ed by F.S. Hill, Prentice Hall (ISBN: 0-02-354856-8), 2001
- Real-time Collision detection by David H. Eberly, Morgan Kaufmann (ISBN-13: 978-1558607323), 2005

E SUGGESTED READINGS AND RESOURCES

1. Courseware (for announcements, quizzes, etc.): `https://edmo.do/j/8uies6`
2. Course Github Organization: `https://github.com/CS179-14-2014-2015`
3. Syllabus: `https://github.com/CS179-14-2014-2015/syllabus`
4. Good Resources for Learning Git and GitHub: `https://help.github.com/articles/good-resources-for-learning-git-and`
5. Git Cheat Sheet: `https://education.github.com/git-cheat-sheet-education.pdf`
6. Ten C++11 Features Every C++ Developer Should Use `http://www.codeproject.com/Articles/570638/Ten-Cplusplus-Featu`
7. C++Now 2014 Presentations `https://github.com/boostcon/cppnow_presentations_2014`
8. Game Programming Gems series
9. Game Engine Gems series
10. Glenn Fiedler's Game Development Articles and Tutorials: `http://gafferongames.com/`
11. The Witness: `http://the-witness.net/news/`
12. Wolfire Games Blog: `http://blog.wolfire.com/`
13. Gamasutra: `http://www.gamasutra.com/`
14. Game Physics by David H. Eberly, Morgan Kaufmann
15. OpenGL Reference: `http://www.opengl.org/sdk/docs/man/`
16. GLSL 1.2 language reference: `http://www.opengl.org/registry/doc/GLSLangSpec.Full.1.20.8.pdf`
17. gDEBugger: `http://www.gremedy.com/`
18. Thinking in C++ 2nd Ed. (Eckel, Bruce): `http://www.mindview.net/Books/TICPP/ThinkingInCPP2e.html`
19. C++ FAQ lite: `http://www.parashift.com/c++-faq-lite/`
20. What Every Computer Scientist Should Know About Floating-Point Arithmetic: `http://download.oracle.com/docs/cd/E19957-01/806-3568/ncg_goldberg.html`

F COURSE REQUIREMENTS

5%	Project Consultation
15%	Class Participation/Recitation
25%	Project
25%	Quizzes and Homework
30%	Final Project Defense/Exam

F.1 Class Participation/Recitation

1. Each recitation is worth 3% (out of the total 15%).
2. Depending on your answer, the grade per recitation can drop.
3. There's no limit on how much you can recite.
4. The recitation will be totaled in the end to compute the class participation grade.
5. You may be called after a homework to present it in class. The presentation would last a maximum of 20 minutes and is worth 4%. The process (architectural decisions, bugs encountered, debugging process and solution) of making the game should be explained. Only the game is needed for presentation; no keynotes allowed.
6. Excess points are spilled over to other components based on the following formula:

$$\log_2(x+1)$$

where x is the recitation points in excess of 15.

F.2 Project

1. Projects will be done by groups of two to three.
2. The output will be an application of the lessons learned in the course.
3. The project need not be original. Just make sure that you cite your source(s), know what you're doing and don't blatantly copy-paste. Remember, the project defense has a heavier weight than your project.
4. The project need not have original art, you may get artwork from resources from the internet. Be sure to cite your sources.
5. Alternatively, you can implement a technique in the suggested readings and resources section. Consult with me first before proceeding with such endeavor.
6. Submit the Group Certificate of Authorship, stating all resources and references used (except for the references listed here), together with the project or during defense. Failure to do so will mean a 0 for the project.
7. Do not plagiarize. Offenders will be dealt severely.
8. Project grade breakdown:

3%	Submission
2%	Defense-time compilation
5%	Proper usage of git
3%	Persistence
3%	"Data-driven"-ness
4%	Polish
5%	Complexity

F.3 Project Consultation

1. The project consultation is an (admittedly futile) attempt to get people moving on their projects.
2. The project consultation is two parts, contributing 2.5% each, the first is to be done anytime before the last of of class in 2014 and the second is to be done before the finals week.
3. Failure to do the consultation before the deadline will result in a 0 for the component.
4. For the first consultation (done before the end of the year), I expect a project plan so I can say whether it's feasible or not.
5. For the second consultation (done before the finals week), I expect a prototype or rough beta of the game.

F.4 Project Defense

Project defenses are 1 hour each. For all intents and purposes, treat the project defenses as comprehensive individual oral exams. You will first be asked to compile your program on the spot, then you may be asked to modify your project on the spot, explain the techniques used in your project or derive equations and formulas discussed in class. Just for information, the more interesting the project is, the greater the chance that I'll ask questions about the project (over asking "classroom questions"). If a question is directed

at an individual, *others may not interrupt; failure to comply counts as not answering the question*. You can have a project defense without a project but I'll be asking hard questions from the topics we've discussed.

There is no dress code for project defenses but at least dress properly.

The project defense will be graded individually on the A to F scale (A = 30%, B+ = 26.25%, B = 22.5% etc.).

F.5 Quizzes and Homework

1. Quizzes are usually unannounced.
2. Quizzes will be given at the start of the class. Latecomers may only catch up.
3. Quizzes are to be written on a A4 bond paper or posted on Edmodo.
4. Although not necessary, non-graphing calculators may be used in quizzes.
5. A decent skill on arithmetic and geometry (including trigonometry) is expected.
6. Expect quizzes when a reading assignment is given.
7. Expect homework to come every week.
8. There will be no quiz on the week when a homework is due except when it is a moved deadline.
9. Homework either culminates the activities during class time or puts the students in a situation where they have to discover things. They are to be done by group.
10. Homework are graded on the scale of 0 to 3.

F.6 Hands-on Exercises

Hands-on exercises are meant to enhance learning by putting theory into guided practice. These are not graded because I believe that grading them while preventing cheating is a futile effort and a waste of time. As such, it is to the student's discretion whether or not to do the hands-on exercises. Be forewarned though that 1) some homework may rely on the exercises and 2) there may be exam questions that are based on the exercises.

F.7 Submissions

All works (homework and final project) should be submitted via Github using the following protocol: (taken from `https://education.github.com/guide/forks`):

1. Fork the original repository (there will be one per homework).
2. Clone the repository to your computer.
3. Modify the files and commit changes to complete your solution (add project comments in the provided README.md file).
4. Push/sync the changes up to GitHub.
5. For any member of the group, create a pull request on the original repository to turn in the assignment.
6. State your group name in the pull request title.

F.8 Late Submissions

The score for late submissions will be multiplied by

$$e^{-0.1x}$$

where x is the number of days, or a part thereof, past the deadline and $e = \lim_{n\to\infty}(1 + 1/n)^n$. For example, if your submission is late by 3 hours and you got 95%, your final score will be

$$95\% \times e^{-0.1(3/24)} = 95\% \times e^{-0.0125} \approx 95\% \times .9876 \approx 93.82\%.$$

No submissions will be accepted past 48 hours before the grading deadline (i.e. they are as good as a 0).

F.9 Bonuses

- Bonus points are given for interesting "extras" added to homework.
- Up to 8% will be credited for contributions to any FOSS projects during the semester.
  - Multiple contributions will be amalgamated and the total bonus shall not exceed 10%.
  - Deadline for notifications of contributions will be on the last day, 23:59, of the finals week.
  - Documentation revisions are considered but will not guarantee bonuses. Do not expect bonuses for trivial corrections such as spelling or grammatical corrections.
  - Pending or unaccepted submissions are considered but will not guarantee bonuses.
  - Wikipedia edits are not considered.
  - For bonuses that are needed to in order pass the class, only game-related contributions are accepted (i.e. one cannot contribute to a Ruby on Rails project to pass the class).
- Bonuses are added to the final grade.
- Bonuses are privileges, not rights. As such, crediting them is up to the instructor's discretion.

G GRADING SYSTEM

[92%,100%]	A	Excellent	[69%,75%)	C	Sufficient
[87%,92%)	B+	Very Good	[60%, 69%)	D	Passing
[80%,87%)	B	Good	< 60%	F	Failure
[75%,80%)	C+	Satisfactory			

Note: $[a, b)$ means a half-open interval that includes a but excludes b. Rounding is done only in the final grade to two decimal places.

H CLASSROOM POLICIES

The class atmosphere will be relaxed but still orderly. The golden rule here is to respect the instructor and not distract others.

1. Usually, thursday is the "lesson implementation" time.
2. Foods and drinks (except water) are prohibited inside the lab.
3. Attendances will not be checked however, they will be noted (i.e. you will be judged).
4. You are responsible for your absences.
5. If you are running late, don't storm or waltz in the classroom.
6. Permission is not needed to leave the classroom but do so discreetly (i.e. don't rage quit).
7. Cellphone usage is allowed during non-exam class times as long as it is used discreetly.
8. Collaboration on homework, including the problem set, is allowed. However, *copying is strictly prohibited*.
9. Those committing academic dishonesty should be ready face the infinity + 1 banhammer from the department.
10. Excessive noise will not be tolerated.
11. Computers and laptops may freely be used. However, I will not repeat lessons due to divided attention brought about by social networking sites and games.
12. Students are expected to learn C/C^{++} on their own. Questions on the language may be asked during consultation hours.
13. No make-up quizzes. No make-up midterms or homework deadline extension unless you were hospitalized, an immediate member of your family died (grandparents included), or you are whisked away due to some competition. Adequate proof must be provided (e.g. medical certificate, ADSA notice) and the instructor must be notified as soon as possible.

I Consultation Hours

By appointment. If you set an appointment with me, keep it or inform me a.s.a.p. if you cannot make it. You may also email me by the email address stated in this syllabus or by my Ateneo email address. Class-related emails sent to any other email addresses will be ignored.