# **GAM200: Project II**

# **C++ 2D Game**

# Fall 2014

**Prerequisites:** GAM150, CS170, CS230, and MAT140

**Schedule:** Thursday 3:30pm—4:50pm (lecture)

Friday 10:00am—12:20pm (first lab)

Friday 12:30pm—3:00 pm (second lab and/or lecture)

**Classroom:** PLATO for lectures, EDISON/TESLA for labs

**Professors:** Benjamin Ellinger, Chris Peters, Rachel Rutherford, and Ellen Beeman

**Contact:** [bellinge@digipen.edu](mailto:bellinge@digipen.edu) (425) 629-5052

[cpeters@digipen.edu](mailto:cpeters@digipen.edu) (425) 629-5053

[rachelr@digipen.edu](mailto:rachelr@digipen.edu) (425) 449-1308

[ellen.beeman@digipen.edu](mailto:ellen.beeman@digipen.edu) (425) 829-4246

**Class Web Page:** The **GAM200** course at [distance.digipen.edu](http://distance.digipen.edu) (join **GameCentral** as well).

**Office Hours: T**ues 3:30pm-6:00pm; Fri 3:00pm-5:00pm (Benjamin Ellinger)

**T**ues 3:30pm-6:00pm; Fri 3:00pm-5:00pm (Chris Peters)

**T**ues & Thurs 1:30pm-3:30pm (Rachel Rutherford)

Tues & Thurs 1:30pm-3:30pm (Ellen Beeman)

*or by appointment*

**Description**

This project is divided into two semesters and focuses on the creation of a simple real-time game or simulation with 2D graphics (3D games are not allowed). Students will work together on teams of three or four members to implement technical features such as audio effects, music playback, pattern movement, simple artificial intelligence, same-machine multiplayer (networking is not allowed), particle systems, scrolling, and simple physics. All projects must be written with a core of C++ code and cannot use middleware such as pre-existing physics engines, networking engines, etc. Additional topics may include basic software architecture, essential development practices, fundamentals of team dynamics, and task prioritization methods.

**Course Objectives and Learning Outcomes**

After completing this course, students will have applied what they learned in their computer science and mathematics courses to a large, team-based project. Students will also learn how to take a basic game design and improve it through iteration and playtesting, how to form strong aligned teams and deal with team conflict, and how to create a flexible code architecture that makes refactoring easy. Specific technical objectives are the application of 2D graphics, 2D animation, parallax scrolling, particle systems, basic 2D collision and forces, pattern movement, simple flocking and path-finding, audio playback, simple menus and HUD elements. After completing this course, students will be ready to move on to a more advanced project in GAM300 or GAM302.

**Textbooks**

There are no required books for this class.

**Optional Textbooks**

C++ For Game Programmers, *by Noel LLopis*, Charles River Media (ISBN: 1584502274)

Game Coding Complete (3rd Edition), *by Mike McSchaffry*, Charles River Media (ISBN: 1584506806)

Game Engine Architecture, *by Jason Gregory*, AK Peters (ISBN: 1568814135)

**References**

You can find the current list of recommended resources on the **GameCentral** page.

**Outline and Tentative Dates**

This class will roughly follow the outline below, although the order and/or content of the lectures are subject to change. The milestones will only be changed in extreme and unexpected circumstances (and will never be moved earlier). Unless stated otherwise in the schedule below, all labs are reserved for meeting with your team, working on your project, or meeting with the instructors.

GAM 200 students must attend all Thursday lectures and the Friday lectures on the first four weeks of class. Friday lectures after the first four weeks are optional for GAM 200 students—you can instead work in the labs if you wish.

**Week 1 (9/2–9/5)**

**Lectures:** Greatness, Iteration, and Team Structure (Thursday); Team Building (Friday, required); Software Development Processes (Friday, required)

**Labs:** Read the syllabus and rubrics. Form teams and talk to instructors if you do not have a team.

**Homework:** Download and go over the game engine sample from **GameCentral**. Discuss with rest of team. No weekly task log is required.

**Week 2 (9/8–9/12)**

**Lectures:** Architecture Principles (Thursday); Data-Driven Architecture (Friday, required)

**Labs:** Questions and meetings. Work out initial team roles and responsibilities.

**Homework:** Design initial engine architecture and begin coding. Set up SVN repository and task/bug-tracking system. Submit weekly task log #1.

**Week 3 (9/15–9/19)**

**Lectures:** Systems, Objects, and Communication (Thursday); Fundamental Development Practices (Friday, required)

**Labs:** Questions and meetings.

**Homework:** Submit weekly task log #2.

**Week 4 (9/22–9/26)**

**Lectures:** Project Management Basics (Thursday); Introduction to Game Physics (Friday, required)

**Labs:** Questions and meetings.

**Homework:** Integrate all systems into core engine. Finalize at least three game concepts. Finalize your team structure. Submit weekly task log #3.

**Week 5 (9/29–10/3) ENGINE PROOF MILESTONE** (individual milestone reports due by midnight Sunday)

**Lectures:** Each team makes the engine proof presentation to the entire class.

**Labs:** Each team makes the engine proof presentation to the entire class.

**Homework:** Write individual milestone reports. No weekly task log is required.

**Week 6 (10/6–10/10)**

**Lectures:** Testing and Usability (Thursday); Modern Sprites, Animation, and Batching (Friday)

**Labs:** Questions and meetings.

**Homework:** Submit weekly task log #4.

**Week 7 (10/13–10/17)**

**Lectures:** Game Pitches and Game Design Documents (Thursday); Meta Classes and Reflection (Friday)

**Labs:** Questions and meetings.

**Homework:** Submit weekly task log #5.

**Week 8 (10/20–10/24)**

**Lectures:** Time Management (Thursday); The Action List (Friday)

**Labs:** Questions and meetings.

**Homework:** Finalize your game pitch and prioritize all tasks needed for your prototype. Submit weekly task log #6.

**Week 9 (10/27–10/31) SEGMENT PROTOTYPE MILESTONE** (individual milestone reports and Game Pitch/GDD due by midnight Sunday)

**Lectures:** Each team makes the segment prototype presentation to the entire class.

**Labs:** Each team makes the segment prototype presentation to the entire class.

**Homework:** Write individual milestone reports and the Game Pitch/GDD outlining the game you will attempt to make. No weekly task log is required.

**Week 10 (11/3–11/7)**

**Lectures:** Personas and Playtest Sessions (Thursday); Details About Playtesting (Friday)

**Labs:** Questions and meetings.

**Homework:** Submit weekly task log #7.

**Week 11 (11/10–11/14)**

**Lectures:** Overview of the Rubric (Thursday); Details About the Rubric (Friday)

**Labs:** Questions and meetings.

**Homework:** Submit weekly task log #8.

**Week 12 (11/17–11/21)**

**Lectures:** Killing the Good, Keeping the Great (Thursday); Test Automation and Logging (Friday)

**Labs:** Questions and meetings.

**Homework:** Finalize the features for your episode prototype and re-prioritize all the tasks needed to get there. Submit weekly task log #9.

**Week 13 (11/24–11/28)** *No classes on Thursday or Friday*

**Lectures:** None

**Labs:** None.

**Homework:** Submit weekly task log #10.

**Week 14 (12/1–12/5) EPISODE PROTOTYPE MILESTONE** (individual milestone reports, TDD, and game submission due by midnight Sunday)

**Lectures:** Selected teams make an episode prototype presentation to the entire class.

**Labs:** Each team finishes their episode prototype and submits it.

**Homework:** Write individual milestone reports, the TDD, and finalize the episode prototype submission. No weekly task log is required.

**Retrieved from "**[**http://www.digipen.edu/main/GAM\_200**](http://www.digipen.edu/main/GAM_200)**"**

**This page has been accessed 2,940 times. This page was last modified 11:48, 25 April 2006.**

**Grading Policy**

The grades for this class are based on the score for the final submission made at the end of the semester. The same base score is given to all members of the same team within the same general discipline (coding, design, art). However, each student has their overall score adjusted by a modifier based on the contributions they personally have made to the project (code, design, art, music, testing, producing, etc.), as determined by their individual milestone reports, along with modifiers for participation, teamwork, attendance, etc. These categories of modifiers are explained below and are a major factor in the final grade. It is possible to get an adjustment of a full letter grade up (and several letter grades down), so it is not necessary for a student’s project score to be an A in order to get an A in the class (and a passing project score does not guarantee that an individual will pass). All grades in this class are given as percentage amounts on the normal DigiPen scale.

Mid-term grades are based on the first presentation and first milestone report of the semester. This will not directly affect your final grade (except for the milestone report part), but is a good indication of whether you are in trouble or not.

***Individual Score = Project Score + Milestone Reports + Coding Reports + Weekly Logs + Attendance + Extras***

**Milestone Reports**

Each milestone, every student in the class must submit an individual milestone report through [distance.digipen.edu](http://distance.digipen.edu/). This report should include everything you have done related to this class since the last milestone. This means coding, debugging, optimizing, designing, testing, art, music, research, organizing meetings, helping other teams, etc. List everything, with an indication about how much time/effort you spent on each item (include the number of hours on each item, even if it is just an estimate). Also include any notes about how the milestone went in general, team problems, praise for others who did something exceptional, or anything else.

Each milestone report will be given a grade from -20% to +5%, but the final total for all three milestones is subject to adjustment (either up or down) based on a more detailed examination of each student’s work at the end of the semester.

* **+5%** (tons of work done this milestone at a very high quality level)
* **+2%** (tons of work done this milestone, or plenty of work done at a very high quality level)
* **+0%** (plenty of work done this milestone)
* **-2%** (moderate amount of work done this milestone, but should have done more)
* **-5%** (some work done this milestone, but much less than should have been done)
* **-10%** (very little work done this milestone)
* **-20%** (no report submitted or almost no work done this milestone)

#### Coding Reports

Each milestone, every student in GAM 200 must also submit an individual coding report through [distance.digipen.edu](http://distance.digipen.edu/). This report must include a detailed breakdown of the lines of code you have personally written so far on the project (this is cumulative, not just what you have done for the one milestone), using a tool like CLOC. You should have well over 400 lines of code by the first milestone, well over 800 lines of code by the second milestone, and well over 1200 lines of code by the last milestone. Code that is not C++ or shader code counts only half as much. This does not include comments, header files, blank lines, etc. It also does not include lots of cut-and-paste code that should really be in functions, code that is just setting variable values by hand instead of reading them from files, etc. Just padding your line count by writing bad code does not meet this requirement. Make sure that multiple team members do not claim the same code (or clearly mark code that multiple team members worked on).

The code you write must also be of sufficient difficulty and complexity (and must work to a decent degree). In general, writing all the graphics code, all of an art pipeline, all of a level editor, all the physics code, all the AI/game logic, or all the core engine code is sufficiently difficult/complex. Most other coding will not qualify, although the instructor can grant exceptions.

Each coding report will be given a grade from -10% to +2%, but the final total for all three milestones is subject to adjustment (either up or down) based on a more detailed examination of each student’s work at the end of the semester. This coding requirement only applies to the first semester of the project, not to later ones.

* **+2%** (not behind on code at all, and code is very difficult/complex)
* **+0%** (not behind on code at all)
* **-2%** (slightly behind on code)
* **-5%** (significantly behind on code, or code done is not difficult or complex enough)
* **-10%** (no report submitted, a milestone behind on code, or code done is very simple)

**Weekly Logs**

Each week, every student in the class must submit an individual weekly task log through [distance.digipen.edu](http://distance.digipen.edu/). This report should list tasks and approximate hours spent on each task. You are strongly recommended to update a work log file every day you work on GAM tasks, rather than writing this report once a week. The weekly work log will not be graded in detail, but failure to submit these assignments is a -1% for each one.

#### Extras

All other elements of the class are considered “extras”. You can get bonuses of +1% to +5% for performing well in leadership roles, helping out other teams extensively, participating consistently and constructively in class, or just doing something exceptional. Skipping a presentation is a -5% penalty.

**Attendance**

Attendance at all labs and lectures is required, although if you email the instructor about any absences, they might be excused (especially if you send the email beforehand, but send one regardless). Each missed lecture or lab is a -2% penalty. Note that absences are counted by the number of attendance sheets that you have not signed. Even if you have photographic proof that you were in class, it does not count if you do not sign the attendance sheet, as this is an accreditation requirement (email the instructor if you forget to do so). Attendance emails go to [ellen.beeman@digipen.edu](mailto:ellen.beeman@digipen.edu) with the subject line of “ABSENCE – GAM 200 – LASTNAME, FIRSTNAME – DATE OF ABSENCE - LAB” (or “LECTURE” or “LAB/LECTURE”). Emails requesting an excused absence that do not follow this format will be denied.

**Late Policy**

If an assignment is turned in late, its grade is reduced by 1% for each day (including weekends) it is late, but never below the grade you would get for not doing it at all. After one week, assignments can no longer be turned in.

Late penalties for the final project of a semester are -5% per day. If a final project requires a resubmission due to major penalties from failed TCRs (which is the only reason a resubmission is allowed), then the grade is reduced by 5% if it is resubmitted within a day of the team being informed of the problem. Each additional day after that results in an additional 5% deduction. If additional resubmissions are required, any additional penalties are cumulative. If your total score for TCRs is -5% or better, you will not even be notified or allowed to resubmit.

**Milestone Rubrics**

Each milestone has a rubric used to score that particular milestone. These are found on Game Central as separate documents. You should read all these rubrics during the first week of class—do not wait until the milestone is actually due. Note that the first two milestones of each semester do not actually affect your final grade in the class. Only the last milestone is used for your final grade. These rubrics and the submission spreadsheet are found on **GameCentral**.

**Pitches and Design Documents**

There are two required documents. The first is the Game Pitch/Game Design Document, which includes information about how your game works, characters, enemies, the setting, the art style, the audio style, etc., all of which is needed to convince a publisher to fund your project. Second is the Technical Design Document, which includes information about how your architecture works, coding methods, naming conventions, and tools (particularly art pipeline and level editing).

High-quality and complete documents give the entire team a +1% bonus, while a really professional-looking document can get a +2% bonus. A further bonus of +1% to +2% can be gained for doing a full game design document as well. Not doing it at all is a -2% penalty (each) for the whole team.

**Producer Responsibilities**

Producers are responsible for driving all scheduling and planning for your project. They create and maintain the sprint/milestone backlog for all coding, content creation, and bug fixing tasks (using scrum boards, TRAC, FogBugz, etc.). The producer holds sprint/milestone planning and prioritization meetings, along with at least one scrum each week. Producers must also attend the weekly producer meetings and convey information to the rest of your team.

Producer must submit weekly status reports (these are separate from and in addition to the individual milestone reports). All other team members are required to give the producer the data for these updates. If a team member does not update their status, just list that as part of the report. Also provide a link to the task/bug list in your status reports (or a screenshot/file attachment if that makes more sense). If you submit all status reports on time for the entire semester, you can get up to a +2% bonus.

**Team Size**

Teams must have at least three members officially getting credit for the project (the team members do not have to all be getting credit for the same class, though). The size of your team will affect your final project grade (i.e., each official team member has a “cost”).

* You start with a +10% bonus to your project grade for team size.
* -2% for each team member getting credit for a regular game project class (GAM/PRJ)
* -1% for each team member getting credit for a non-game project class (CG/GAT)
* -0% for each team member who is not getting credit, or who is a sound designer in a MUS class
* If a student is only on a team for part of a semester, their cost is split among the teams they were/are on.
* The team size bonus can go negative, but then it becomes a penalty to your project grade.

**Team Structure**

Team structure must be determined by mutual agreement among all team members. The only hard requirement is that one student must assume the role of producer, who will be the team’s primary contact person with the instructors. While there are many other possible roles (see below), only the producer role is required. Note that being your team’s producer does not get you out of the other requirements for the class (coding, designing, etc.).

There are many different structures you can choose among to organize your team, each with its advantages and disadvantages. There are two parts to this: leadership roles and developer roles.

Leadership roles include titles such as Producer, Director, Technical Lead, Design Lead, Art Lead, Audio Lead, Test Lead, etc. Traditionally organized teams have all of these roles, with the Producer being responsible for planning and organization, the Director being responsible for the overall direction and creative vision of the game, and the various leads managing developers of their specific type (but ultimately reporting to the Director). This works best for large teams where having leads for each discipline is important. For smaller teams, having leads might not make sense, or might only make sense for certain disciplines. In many cases, the Director may also be a lead (usually a Technical, Design, or Art Lead) or a Producer, especially if the team is small. Some teams do not even have a Director—instead the leads act jointly as a director (this can easily weaken a game’s vision if you are not careful, though). At the most extreme are teams that have no formal leadership roles at all (Valve does this). This can work with just the right team, but can also make it difficult to ship a game on time (Valve’s main weakness).

Developer roles are divided up by the actual work that needs to get done. This can just be generic, by discipline, so that the roles are Programmer, Designer, Artist, Musician, Tester, etc. It can also be more specific, such as Graphics Programmer, Physics Programmer, Networking Programmer, Systems Designer, Level Designer, Animator, Concept Artist, Modeler, Composer, etc. While a single developer usually has only one specific role on a large project, smaller projects tend to have developers fulfilling multiple roles at the same time. Having very specific roles can be good because it makes it very clear what each developer is responsible for, but it can be very inefficient and cause problems when, for example, only one developer knows how the graphics engine works. At the most extreme, we find Valve again, which doesn’t even have formal developer roles (they have developers who do both art and code, for example). This works because they are very careful about the type of people they hire, but many developers would get lost without a clear idea of what their role on the team is.

DigiPen teams have succeeded with just about every type of team structure there is. Your team must determine what structure it is going to use and make that a part of the first milestone presentation.

**Mechanisms and Procedures**

There are a variety of procedures and mechanisms used in this class to make it run as smoothly as possible. Make sure you read each of these sections thoroughly so that you understand what the instructors expect.

**Instructor Questions and Meetings**

You will undoubtedly have many questions for the instructors and will often wish to have individual or team meetings as well. To make this work efficiently, you must email any questions (about any topics you wish) or meeting requests to one of the instructors. Make sure you start the subject of the email with “GAM200”, “GAM300”, etc. so that it won’t be filtered out (failure to do so will result in unanswered emails—note that it must be capitalized and have no spaces or dashes).

In addition to asking questions through email, if you talk with an instructor in person (whether in class or otherwise) and there is some follow-up action the instructor has agreed to perform, you must email that instructor with a reminder. If you don’t send a follow-up email, whatever you talked about will be forgotten and not followed up on (regardless of what the instructor said at the time). Making follow-up emails a habit is excellent practice for the real-world of working with busy bosses, producers, executives, etc.

**Team Names**

Students can generally select any team name they wish, so long as it would be appropriate for an E-10 rated game. However, the official name of the team for competitions and/or press interactions must be DigiPen ***teamname*** (or similar wording). For example, the official name of “Team Nitronic” would be “DigiPen Team Nitronic”. Note that team logos (even ones that include the team name), do not have to include “DigiPen” as part of them.

**Game Competitions**

DigiPen games can only be submitted to competitions by the DigiPen faculty—you cannot enter them yourself. If you think you have a game good enough to be entered into competitions (or that is the goal you are aiming for), make sure you inform your instructors as soon as possible, as they can give you advice directly targeted at making your game better for competitions. As general rule, any game that doesn’t get a final score of 90% or higher will not be submitted to a competition (unless it is later improved). To have a real chance of winning a competition, you’ll usually need a 95% or more.

**Changing Teams**

Students can leave or change teams with no penalty. You can kick someone off your team with a unanimous vote by the rest of the team. You must talk to an instructor about this, but they will not overturn your decision. If you leave your team and cannot find a new one, then you will be placed on the “studio team”, which is run by one of the instructors. Solo projects are not allowed.

If you do not stay on the same team for the entire semester, your grade is broken up into thirds (1st milestone, 2nd milestone, 3rd milestone) and each third comes from the final score of the project you worked on during that milestone. If you changed teams in the middle of a milestone, your grade for that milestone will be from the new project you are working on.

**Game Content**

DigiPen games must be able to get an EC, E, or E10+ ESRB rating. Anything that would require a T (13+) rating requires permission from an Assistant Dean. M (17+) and AO (18+) ratings are not allowed under any circumstances.

*Violence:* only cartoon/fantasy violence is allowed—no gore, body parts, realistic blood, etc.

*Social Issues:* any references to real-world politics or alcohol/tobacco/drugs require approval.

*Sexual Content:* nudity, sex, strongly suggestive sexual themes or references are not allowed.

*Language:* profanity and disparaging/stereotyping of race/gender/culture/disability are not allowed.

Remember that all art and audio must either be created by a current DigiPen student/instructor or be from the DigiPen approved art and audio libraries. You cannot use your friends, family members, public domain material, or other students not in your class (unless you talk to your instructor first). You can never use outside artists/musicians at all.

**Source Control**

Each team is responsible for setting up and managing source control for their project. See <https://svn.digipen.edu/> for details (Mercurial repositories are also available). Contact the IT department to get a repository for your project. You must use DigiPen’s source control, not an external server.

**Technical Restrictions**

You must code everything yourself in C++ (C# cannot be used), using only fairly low-level libraries (GDI, DirectX, OpenGL, OpenAL, FMod, STL, SDL, and XML parsers—if you want to use anything else, you must ask first). You can use scripting languages (Lua, Python, etc.) that you embed in your C++ engine. Physics, AI, and networking libraries are not allowed.

Your game cannot be 3D or be networked. If you want to use 3D graphics and/or networking, you can have modes in your game for these features, but those modes will not be considered for grading purposes (nor will any work on those modes be counted for milestone report purposes). If your game is partially 3D in some way and you are not sure whether that counts, check with your instructor to see if it is allowed.

**Last Day to Withdraw**

In order to withdraw from a course it is not sufficient simply to stop attending class or to inform the instructor. In accordance with the policy, contact your advisor or the registrar to begin the withdrawal process. The last day for withdrawal from this course is cited in the official catalog.

**Academic Integrity Policy**

Cheating, or academic dishonesty in any form, will not be tolerated in this course. Penalties for cheating may include receiving a zero on an assignment, or a failing grade in the course, or even expulsion from DigiPen. For further details, please consult the *DigiPen Academic Integrity Policy*. Note that in this team project class, working directly with your teammates, or even with other teams, is not cheating (and is highly encouraged). However, each student is required to accurately inform the instructors of the exact work they personally did on the project—any deception is cheating and will be punished harshly.

**Disabled Student Services**

If students have disabilities and will need formal accommodations in order to fully participate or effectively demonstrate learning in this class, they should contact the Disability Support Services Office at [(425) 629-5015](mailto:(425)%20629-5015) or [dss@digipen.edu](mailto:dss@digipen.edu). The DSS Office welcomes the opportunity to meet with students to discuss how the accommodations will be implemented. Also, if you may need assistance in the event of an evacuation, please let the instructor know.