COMP 2501A (Winter 2025)

Computer Game Design and Development

Instructor: Oliver van Kaick

Gender pronouns: he/him/his (learn more)

Email: Oliver.vanKaick at carleton.ca

Office Location: HP5348, Herzberg Building

Best Ways to be in Touch: in class, during student hours, via the course's Discord server and Discord DMs, or via email

Student hours: student hours will be listed in Brightspace once the course starts.

Teaching Assistants: A list of teaching assistants and their contact/office hours information will be available in Brightspace once the course starts.

Class Location: Please check Carleton Central for the room location.

Lecture Times: Mondays and Wednesdays, 2:35pm to 3:55pm

Tutorial Times: Mondays 11:35am-12:55pm (A3), Mondays 1:05pm-2:25pm (A5), Tuesdays 8:35am-9:55am (A4), Tuesdays 10:05am-11:25am (A1), Fridays 8:35am-9:55am (A2)

Tutorial Location: HP5151

Course Website: https://brightspace.carleton.ca/d2l/home/283322

Course Discord server: you can find an invitation to join the server in Brightspace.

Important dates and deadlines can be found here: https://carleton.ca/registrar/registration/dates/academic-dates/, including class suspension for fall, winter breaks, and statutory holidays.

Course Summary

COMP2501 is an introduction to the practical development of computer games. The course covers a variety of mathematical concepts, algorithms, and software technologies relevant for the creation of 2D games. The course includes programming assignments and a project which consists in the development of a computer game.

Course Calendar Description

Introduction to the practical development of computer games and engine architecture. Topics include: vector and matrix operations; coordinate systems and transformations; physical simulation; collision detection; AI; path planning; hardware-accelerated real-time rendering. Special attention is given to implementation of real-time rendering in a low-level language.
Includes: Experiential Learning Activity
Prerequisite(s): COMP 1501, COMP 2401 with a minimum grade of C-, and MATH 1104.
Lectures three hours a week, tutorial one and a half hours a week.

Learning Material(s) and Other Course/Lab-Related Resources

Students are not required to purchase textbooks or other learning materials for this course.
We do not have an assigned textbook for the course. I recommend using Sanjay Madhav's *Game Programming: Algorithms and Techniques* as a reference for reviewing the different topics covered in the lectures. For the assignments and the course project, we will be programming in C++ and using a set of free libraries that build on OpenGL. The libraries will be posted on Brightspace. For detailed questions on programming with C++ and OpenGL, there are a wealth of books, websites, and online tutorials that provide information; a few recommendations of free online resources are provided in the Brightspace page.

Learning Material	Options for Purchasing	Approximate Cost (New)
Sanjay Madhav's *Game Programming: Algorithms and Techniques* (optional)	Amazon, used	CAD $73

Topics Covered

- Game architecture: MVC design pattern, game object management
- Mathematical foundations: vector operations, coordinate systems, and transformations
- Introduction to hardware-accelerated real-time rendering: geometry and shaders
- Introduction to OpenGL
- Physical simulation and collision detection
- Game AI and path planning

For a detailed list of the topics covered per week, please refer to the calendar available in Brightspace.

Learning Outcomes

At the end of this course, students will be able to:

- Summarize the main components necessary for the development of a computer game based on 2D graphics and physical simulation.
- Explain the principles behind the fundamental techniques used for the creation of 2D games (the topics listed above), discussing the mathematical operations and algorithms involved in these techniques.
- Identify the most suitable techniques to create specific features in a 2D computer game.
- Implement a basic 2D game in C++ with OpenGL graphics and auxiliary libraries.

Assessment Scheme

Grading scheme (the specific deadlines can be found in Brightspace):

Component	**Percentage**	**Date**
Assignments	35%	Approximately every two weeks
In-class exams	15%	Two exams during class time, around February and March
Course project	15%	Due at the last day of classes
Final exam	35%	Scheduled centrally, during exam week

Note that you need to obtain a passing grade in the exams to pass the course. The exams include the in-class exams and the final exam.

Late and Missed Work Policies

Assignment deadlines are strict. The following scheme is applied to late submissions (which includes assignments and the final course project):
- 3 hours late: no penalty
- 3 to 12 hours late: -10%
- 12 to 24 hours late: -20%
- More than one day late: assignment receives a grade of zero.

If you encounter extenuating circumstances that prevent you from submitting the assignments on time, read more about academic consideration forms and long term accommodation.

Assignment submissions are handled electronically (i.e., through Brightspace). Technical problems do not exempt you from submitting on time. So, if you wait until the last minute and then have issues with your connection, you will receive a deduction according to the scheme above. Consequently, you are advised to:

- Periodically upload you progress (e.g., upload your progress to Brightspace after each major change; we will only grade your last submission).
- Submit your final work at least one hour in advance of the due date and time.
- Store backups of your assignments in the cloud, e.g., OneDrive, Dropbox, a private GitHub repository. However, your assignment has to be submitted to Brightspace so that we have a timestamped, archived submission of your work. Urls to the cloud will not be accepted.

The assignments consist of programming tasks. If any of the assignments that you submit does not compile or run, it will receive a mark of zero. Consequently, after you upload your submission to Brightspace, you should re-download it immediately and ensure that the project can be created with cmake, compiled, and run.

You are expected to demonstrate good programming practices at all times and your code may be penalized if it is poorly written. You are also expected to do the necessary preparatory work (i.e., devising an algorithm) before you start coding.

Academic Integrity and code reuse

Students are expected to uphold the values of academic Integrity, which include fairness, honesty, trust, and responsibility. Examples of actions that compromise these values include but are not limited to plagiarism, accessing unauthorized sites for assignments or tests, and unauthorized collaboration on assignments or exams.

You are free to make use of art assets found online provided that their license allows you to freely use the assets and you credit the source. Code fragments that are not of your own authorship are allowed under the following conditions: 1. The code should not be implementing the main tasks required for an assignment, but rather serve for adding additional features to the project; 2. Provide credit to the original author of the code and make sure that you understand what the code is doing.

If you are unsure about the expectations regarding academic integrity (how to use and cite references, how much collaboration with lab- or classmates is appropriate), ask your instructor. Sharing assignment specifications or posting them online (to sites like Chegg, CourseHero, OneClass, etc.) is considered academic misconduct. You are never permitted to post, share, or upload course materials without explicit permission from your instructor.

Misconduct in scholarly activity will not be tolerated and will result in consequences as outlined in Carleton University's Academic Integrity Policy. A list of standard sanctions in the Faculty of Science can be found here.

Additional details about this process can be found on the Faculty of Science Academic Integrity website.

Students are expected to familiarize themselves with and abide by Carleton University's Academic Integrity Policy.

Use of tools based on artificial intelligence

All of the assessed activities in this course except for the project were designed to be completed by an individual working alone. The use of artificial intelligence tools such as ChatGPT, Copilot, etc., is allowed in programming assignments. However, if you use any of these tools, you have to disclose this in a radme.txt file or report when submitting the activity, just as you would do when reusing code from other sources. Explain what portions of the assignment were created solely by you and what portions were created with the aid of the tool. The use of any of these tools without disclosure will be considered academic misconduct. An exception to this rule is made for automated grammar and punctuation checking tools (such as Grammarly). In addition, you are fully responsible for your code, whether you produced it with AI tools or not. Do not fully trust the output of AI tools and double check any generated code, as AI tools are known to produce incorrect outputs.

School of Computer Science Laptop Requirement (only applies to on-campus courses)
Every student that has been enrolled in a 1000-level (i.e., first year) course offered by the School of Computer Science after the 2020/2021 school year is required to have a laptop. This includes COMP1001, COMP1005, and COMP1006. For more information, please visit https://carleton.ca/scs/scs-laptop-requirement/ and then review the requirements at https://carleton.ca/scs/scs-laptop-requirement/laptop-specs/.

Student Rights & Responsibilities

Students are expected to act responsibly and engage respectfully with other students and members of the Carleton and the broader community. See the 7 Rights and Responsibilities Policy for details regarding the expectations of non-academic behaviour of students. Those who participate with another student in the commission of an infraction of this Policy will also be held liable for their actions.

Undergraduate Academic Advisors

The Undergraduate Advisors for the School of Computer Science are available in Room 5302HP; or by email at scs.ug.advisor@cunet.carleton.ca. The undergraduate advisors can assist with information about prerequisites and preclusions, course substitutions/equivalencies, understanding your academic audit and the remaining requirements for graduation. The undergraduate advisors will also refer students to appropriate resources such as the Science Student Success Centre, Learning Support Services and Writing Tutorial Services.

SCS Computer Laboratory

Students taking a COMP course can access the SCS computer labs. The lab schedule and location can be found at: https://carleton.ca/scs/tech-support/computer-laboratories/. All SCS computer lab and technical support information can be found at: https://carleton.ca/scs/tech-support/. Technical support staff may be contacted in-person or virtually, see this page for details: https://carleton.ca/scs/tech-support/contact-it-support/.

Mental Health and Wellness

Visit the Carleton Wellness Website for resources.

Academic Accommodations and Regulations

Academic Accommodation

Carleton is committed to providing academic accessibility for all individuals. You may need special arrangements to meet your academic obligations during the term. The accommodation request processes are outlined on the Academic Accommodations website (https://students.carleton.ca/course-outline/).

Student Concerns

If you have any concerns regarding this course, your first point of contact is me. Please email me or visit during my student hours, and I will do my best to address your concerns. If I cannot resolve the issue, the next point of contact is the School of Computer Science at studentconcerns@scs.carleton.ca. If the concern remains unresolved, the final point of contact is the Office of the Dean of Science at ODScience@carleton.ca. Please follow this order of contact. Note that you can also bring your concerns to Ombuds services.