CS529: Fundamentals of Game Development

Fall 2011

Prerequisite:	Programming: Students must be proficient in C/C++ Math: Knowledge of elementary algebra, trigonometry, and elementary calculus (e.g. dot products)
Schedule:	Monday/Wednesday: 4:30pm – 5:50pm
Classroom:	CURIE
Instructors:	Antoine Abi Chakra, Chris Peters
Contact:	Email: achacra@digipen.edu (Specify CS529 in the email subject field) Phone ext.: 5029 Email: cpeters@digipen.edu
Class Webpage:	https://distance.digipen.edu is the online repository for CS 529. Students registered for CS 529 are automatically enrolled in the online version of the course. Please post questions regarding class material and assignments in the forums on the course' web page.
Office Hours:	Antoine: Tuesday/Thursday: 3:00pm – 5:00pm

Description:

Fundamental of Game Development [CS 529] is divided into two main parts.

The 1st part presents foundational data structures, algorithms, mathematical concepts and techniques used in the design and development of two-dimensional real-time interactive simulation and game software. Topics covered include event-driven programming, game engine design and architecture, real-time rendering, user interaction, state machines, animation techniques, collision detection, space partitioning and particle systems.
Several games will be implemented, taking advantage of the aforementioned topics.

The 2nd half of the course will cover game object systems and game architecture. First covering high level architecture explaining object oriented programming, component based design and functional design. Secondly, explaining data driven architecture, serialization, data base design and data inheritance. Thirdly, focusing on communication with messaging, events and signals/slots. Finally, there will be a general discussion of different game architectures and Q/A on design.

Course Objectives and Learning Outcomes:

Upon successful completion of the course, students will learn

- Foundational topics in programming interactive applications including simulations and games.
- Gaining programming experiences in C/C++ and getting familiar with using game engines.
- Ability to design and implement simple 2D games with special emphasis on:
    - Game loop
    - Frame rate controller
    - Basic collision-detection techniques
    - State machines
    - Applications of kinematics and dynamics in game physics
    - Particle systems
- How to handle the "time factor" which is a main component in all time based applications and games, and the synchronization between the game loop time and the "real" time within the game loop, which is extremely useful in physics formulas and simulations in general.
- Starting point for understanding technical literature and foundation for independent research related to game programming and development.
- Implementation of mathematical techniques including linear algebra.
- Game Engine Architecture
- Large Scale Engine Architecture
- Events and Communication
- Data Driven Design
- Real time Physics
- Real time graphics

Textbook: None required

Outline and Tentative Dates:

Please note that this is a tentative organization of the course and may be subject to change.

Introduction to Game Programming and Game Engines – 1 Week

- RTIS
- Overview of 2D/3D Game Engine Architecture and Design
    - Graphics Manager
    - Simulation Manager

  - Collision Detection Manager
  - Streaming File I/O Manager
  - Audio Manager
  - Game State Manager
  - Scene Management

Engine Flow Implementation – 1 Week

- Game State Manager
- Frame Rate Controller
- Overview of Event-Driven Programming
- Win32 Programming

Engineering Non-Interactive Simulation – Cage – 1.5 Weeks

- Review of Vector Algebra and Geometry
- Static Collision
- Normal Line Equation
- Collision Detection Techniques
  - Intersection Between Animated Point and Finite Line Segment
  - Intersection Between Animated Circular Object and Finite Line Segment
  - Intersection Between Animated Circular Object and Stationary Circular Object
  - Intersection Between Animated Circular Objects
- Reflection
- Space Partitioning Techniques

Engineering a Complete Interactive Game – Asteroids – 1.5 Weeks

- Reading User Input
- Transformation Matrices
- Implementing Object Dynamics
  - Acceleration, Deceleration and Friction
- Runtime Object Creation (e.g. Bullets)
- More Collision Detection Techniques
  - Intersection Between Animated Rectangles
- Implementing Gameplay Aspects (Scoring, Game flow…)

Introduction and Implementation of Advanced Topics – 0.5 Weeks (Tentative)

- Memory Manager
- Particle System
- Random Numbers

Engineering Scrolling Platform Game – Platformer – 1.5 Weeks

- Collision Detection Techniques for Platform Games
  - Binary Collision Detection
- Implementing Object Dynamics
  - Projectile Motion – Jumping
- Implementing State Machine Based Behaviors (AI)

Lab time (2nd half of the semester) will be split between lecture and working on project.

Game Engine Architecture – 1 Week

Large Scale Engine Architecture – 1 Week

Mid Point Check In and Milestones – 1 Week

Events and Communication – 1 Week

Data Driven Design – 1 Week

Real Time Physics – 1 Week

Real time Graphics – 1 Week

Rubrics and Assessment:

Grades will be derived from projects, exams and class participation.

- Project 1: 10% - Game State Manager
- Project 2: 13% - Cage
  - Part 1: 6.5% - Bouncing Ball
  - Part 2: 6.5% - Cage (Circular Pillars & Animated Circles)
- Project 3: 12% - Asteroids
  - Part 1: 6% - Vectors & Transformation Libraries
  - Part 2: 6% - Gameplay

- Project 4: 15% - Platformer
    - Part 1: 5% - Collision Functionalities
    - Part 2: 10% - Uniform Grid, Map Transformation, Main Character, Enemy AI, State Machine
- Final Project: 40%
    - Note: The final project will cover the 2nd part of the course. Check the course description for more information.
    - Detailed grading policy can be found at the bottom of this syllabus.
- Attendance and Class Participation: 10%

Grading Policy:

The detailed assessment guidelines for each project will be provided on the class website when the project is assigned. Please make sure to read the document before you start the implementation.

Late Policy:

- Assignments are due at the specified time on the specified due date that will be published on the class website when each project is assigned. More precisely, any programming work submitted ***after the specified time on submission day will be considered late. Late assignments will be penalized 15% for each day (including weekends and holidays) beyond the deadline.***
- Submission procedures for programming assignments are detailed below.
- ***Failure to follow any of the procedures and guidelines specified in this document will result in deductions from your assignment grade.***

Last Day to Withdraw:

In order to withdraw from the course it is not sufficient simply to stop attending the class or to inform the instructor. In accordance with the policy, contact your academic advisor or the Registrar to begin the withdrawal process. The last day for withdrawal from the course is cited in DigiPen's official website.

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 a project, a failing grade in the course, or even expulsion from the school. For further details, please consult the *Digipen Academic Integrity Policy* in the school's catalog.

Assignments and Academic Integrity

CS 529 assignments consist of programming assignments. Assignments are **NOT** group projects. They must represent a student's own individual work. It is reasonable for students to consult or discuss general solutions to an assignment. However, it is unreasonable for students to collaborate on detailed solutions, to copy code, or to give away code. Refer to the Academic Integrity Policy outlined earlier in this document for the resulting course of action for any instance of academic dishonesty.

Assignment Specifications

Each assignment will consist of the following sections:

- **Topics**: Specifies the list of topics covered by the assignment.
- **Programming Statement**: Specifies the list of C/C++ functions required to complete the programming portion of the assignment.
- **Deliverables**: Specifies the list of files and documentation to be submitted.
- **Objectives**: Specifies the list of objectives on which assignment submissions will be graded.
- **Submission Deadline**: Specifies the time and date at which the assignment is due.

Program Submission

The guidelines to be followed in implementing and submitting your program are specified in the following sections:

- Your programming submission must be written using C++ under Visual Studio 2005 or above.
- Your programming submission must be a Win32 application.
- Your programming submission may use Standard C/C++ library.
- Your programming submission must contain a **README** text file. Simply said, the README file tells us how to run your program, how to use parts of your user interface that are not specified in the assignment description, and any assumptions about how we must interact with your program that, if violated, might cause your program to fail. Other useful information that this file should indicate are the hardware, operating system, settings, etc., you compiled, linked, and tested the program in.
- The README, source, header, and data files must start with the header specified here:

```
/* Start Header -------------------------------------------------------

Copyright (C) 20xx DigiPen Institute of Technology.
Reproduction or disclosure of this file or its contents without the prior
written consent of DigiPen Institute of Technology is prohibited.

File Name:        <put file name here>
Purpose:          <explain the contents of this file>
Language:         <specify language and compiler>
Platform:         <specify compiler version, hardware requirements, operating
                  systems>
```

```
Project:          <specify student login, class, and assignment. For example:
                  if foo.boo is in class CS 529 and this file is a part of
                  assignment 3, then write: CS529_fooboo_3>
Author:           <provide your name, student login, and student id>
Creation date:    <date on which you created this file>

- End Header ----------------------------------------------------------*/
```

- To submit your programming assignment, organize a folder consisting of ALL relevant source code (including source files, header files, data files), documentation files, Visual C++ workspace and project files. In other words, *your submission must be ready for compiling and linking by the grader*. Name this folder using the following convention:

  <class>_<student login name>_<assignment#>_<part#>

  For example, if your login is ***foo.boo*** and assignment 1 part 1 is being submitted, your folder would be named **CS529_fooboo_ assignment1_Part1.zip**
- Your folder must not contain Debug and Release folders, object or executable files.
- Zip this folder and name the resulting file using the following convention:

  <class>_<student login name>_<assignment#>_<part#>.zip

  For example, if your login is ***foo.boo*** and you are submitting assignment 1 part 1, your zipped file would be named as: **CS529_fooboo_ assignment1_Part1.zip**
- Next, upload your zip file after logging into the course web page using the link https://distance.digipen.edu.

Finally, perform a sanity check to determine if your programming submission follows the guidelines by downloading the previously uploaded zip file, unzipping it, then compiling, linking, and executing your submission.

Masters Game Engine Technology Project

Grade	F	D	D	C-	C	C+	B-	B	B+	A-	A	A
Percent	50%	60%	65%	70%	73%	77%	80%	83%	87%	90%	95%	100%

Grading Details: Your grade starts at 60%.

- Debug Drawing
  - Basic Debug Drawing – Simple Lines and bound volumes. (Missing -10)
  - Advanced Debug Drawing – Text, duration, or other features +2
- Object Architecture
  - Basic C++ Objects (Missing -20)
  - Component Based or Other Advanced Design +4
  - Game Object Ids, Garbage Collection or some form of object management +2
- Communication
  - Basic Events (Missing -10)
  - Function Binding +2
  - Signals / Slots +2
- Data Driven Design
  - Basic Text Serialization. (Missing -20)
  - Creation of objects from data files. +2
  - More advanced serialization or multiple formats. +2
  - Level Files +2
  - Archetypes +2
  - Level Editor +4
- Graphics
  - Hardware accelerated sprite rendering (textured quads in OpenGL or DirectX) (Missing -20)
  - Basic Sprite transformation (Translation, Rotation, and Scale) (Missing -20)
  - Sprite Animations +2
  - Sprite Z Sorting + 2
  - Sprite Batching +2
  - Other advanced graphics features (Any +2, Max +6)
- Physics
  - Basic collision detection between circles (Missing -20)
  - Basic Collision response (bouncing etc) (Missing -10)
  - Impulse base collision response +2
  - Collision between different bodies type (circle, quad) +2

    - Advanced physics (rotations, constraint based, stacking, etc) up to +6
- Game Requirements
    - Human player control of a ship, avatar, or character. (Missing -20)
    - Game runs at 30 fps most of the time (Missing -10)
    - At least 2 different types of enemies with different behaviors.
        - Additional enemy behavior +2 (max 6)
    - Player has 2 different types of weapons or power ups.
        - Addition weapon type +2 (max 6)
    - Collision between projectiles, player, and enemies. (Missing -20)
    - Controls screen (Missing -20)
        - Menus +2
    - All the requirements must be easy to test including debug features. If I can not find the feature it will be considered missing.
    - Game must have a win / lose condition. (Missing -20)
    - Game must be in C++
    - Game must run on Lab machines.
    - Game is fun for a few minutes (or funny, interesting, surprising)+4
    - **Every crash or soft lock -2**
    - Other Features
        - Impressive features not listed above—see instructor.