Computer Science
2nd Year Games Design
(CSC2003S)

1. Introduction

Welcome to the Computer Science Games Design course for 2012. This document outlines the practical hand-ins for this course. This year, the practicals all follow on from one another. At the end of the course you will have a fully featured 2D game. You will be provided with a simple games engine, from which you build your games. Instructions are included with regards to setup information. Game development will be completed in a Java environment and it is suggested that you make use of a Java IDE. You will be given the installation files for the Eclipse Java IDE but are free to make use of any alternatives.

If you have any queries feel free to contact your assigned tutor or the TA if the tutors are unable to assist you.

2. Overview

The game engine that is provided for you to use in these practicals provides you with the tools to create a 2D top-down shooter. A top-down shooter is a game in which you control the main character directly from an aerial viewpoint. A good example to look at would be CrimsonLand or the sample game provided; a link to CrimsonLand will posted on Vula.

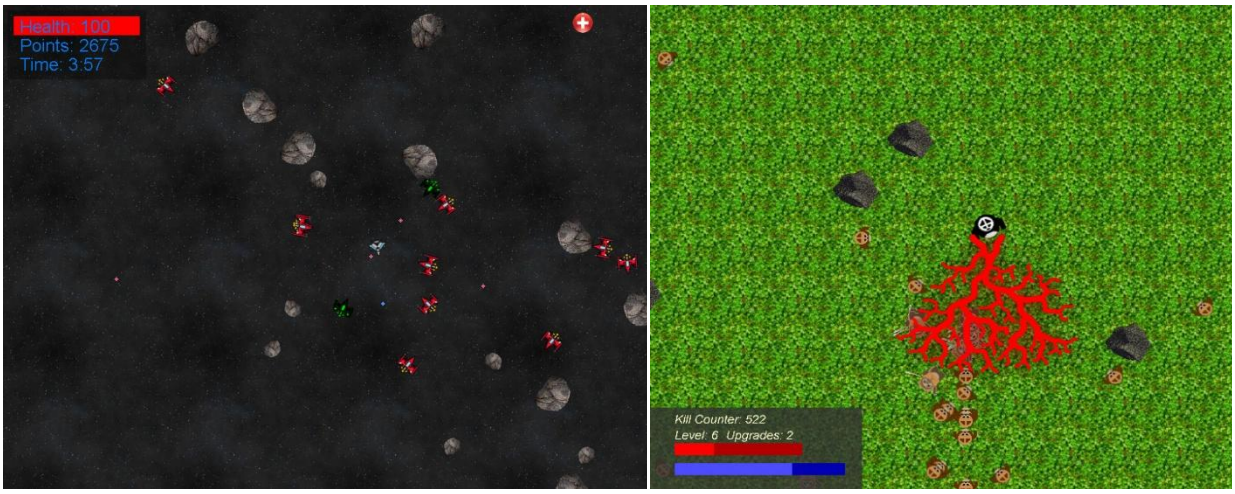

Figure 1: Sample games from the class of 2009

You will be given a package (.zip) which contains all the files you will require, this includes some source files and textures which will generate a sample bare-bones game. You will build your game from this sample. You are required to add additional textures to fit your chosen games theme. You will get a .jar file which contains the compiled game engine which you will use as the game API. You will also get the JOGL libraries and JavaDocs explaining the game engine API calls. It is imperative that you read these documents as soon as possible, to establish familiarity with the API.

You are required to create a top-down shooter-style game similar to those shown above, implementing both controllable movement and the ability to shoot some sort of projectile. Try steer away from direct "CrimsonLand" clones and research some more imaginative options. Try spending some time playing some old Nintendo ROMS or other 2D games for inspiration.

The actual practical explanations later on in this document will detail the specific requirements that you need to implement. These practicals focus on the technical aspect of games development. You will not be marked based on the visual aesthetics of your games but rather the technical difficulty. You are free to create any game, provided you make use of the game engine and it can run on the Senior lab machines.

3. Resources

You will be provided with some resources in order to complete these practicals. These files can be found under the “2D Games Prac” folder listed on the Vula sites resources section. These files include the following.

3.1.Development Tools

The IDE’s that can be installed and links to their websites. The Java development and runtime installation files will also be provided. Eclipse and NetBeans will be provided both for Linux and Windows platforms. Eclipse will be favoured as only Eclipse install instructions are provided. If you are unsure, stick to Eclipse.

3.2.Games Engine

This is where you can get a copy of the “2DGame_x.x.jar” files which will be provided for variety of platforms (Windows & Linux) in 32/64-Bit to accommodate everyone. These files will be distributed in a zip file which contains the game engine, documentation for the engine and some textures to get you started. Please indicate in your readme files which version you are using.

4. Marking guide

Marking of the practicals will be conducted by the tutor to whom you were assigned. A breakdown of the mark allocation will be provided in each of the practical specifications which are the same that the tutor will be given to mark with. Practicals will be marked according to this scheme, so pay attention to them. The work will be cross-checked with other student’s submissions to check for plagiarism as well as checking for code taken from the internet. Practicals are usually marked within the first week following the hand-in and at latest will be marked within 2 weeks.

A Readme file **MUST** be provided in all of the coding practicals. It should contain a list of the game controls, a description of the implementation and any other important information about your code. This is to aid the tutors in compiling/running your game and to highlight any implemented features for marking purposes. Additionally, this file should contain a How To/Key Assignment map to facilitate quicker testing. A penalty of 10% will be applied if a Readme file is not included in any of the coding assignments.

Marks are also awarded for coding style according to the following criteria.

4.1.Consistent layout

This applies to indentation, use of brackets, use of whitespace and the appropriate style of identifiers.

4.2.Appropriate user defined identifiers

Names that make the code easier to read and understand.

4.3.Comments

Comments are vital during the marking process to explain complex pieces of code that cannot be understood at a quick glance. Also do not comment for the sake of commenting as this makes it difficult to read.

4.4.Object/Data Orientated Structure

By now you should know to design programs with an object orientated structure. Do not put everything into one file. Data orientated structure (not recommended for the beginner) is acceptable but should still use encapsulation.

5. Submitting practicals

All practicals must be submitted on Vula by 10:00 am on the morning that the practical is due. Please note that some of the practicals might be fairly large and will take some time to submit to Vula, so please do not wait until the last minute to try submitting. If for some reason you are unable to submit the file to Vula then contact your tutor immediately to arrange for an alternative hand-in method, it is your responsibility to get the practical handed in.

In your submission you only need to submit the source code, a Readme explaining your implementation and any other necessary content like textures. You do not need to submit the JavaDocs, JOGL libraries or the game engine “2DGame_x.x.jar” file as these are quite large and the tutors will have local copies. You must include a Readme file which outlines the details of your practical, including the game controls and practical description along with any special setup requirements that need to be done by the tutor to get the game running.

6. Compiling of practicals

Please make sure that the practicals you submit can be compiled on the senior lab computers and should there be any additional libraries or dependencies, please state as such in your readme files. You will receive a penalty of up to 50% of your final mark, depending on the severity of the compilation issues. Should any issues arise you will be contacted by your assigned tutor and if you are certain that your code compiles correctly you may opt to demo your code in the senior lab. In this case you must provide the code used in the demonstration to the tutor immediately afterwards for the tutor to verify that it is the same as the Vula submission.

7. List of practicals

Here is the list of all the practicals that you will be completing this year with their start and hand-in dates. Please note these dates and make sure that you submit them on time. Practials must be handed in by 10h00 on the morning of the hand-in date.

Prac	Wks	Tutorial Name	Start	Hand in
0	1.5	Game Maker Practical	Fri, 27 July	Wed, 8 Aug
1	1	Game design & tech spec doc	Mon, 13 Aug	Mon, 20 Aug
2	2	Collision detection & bit-checking	Wed, 22 Aug	Mon, 10 Sept
3	3	Game features implementation	Mon, 10 Sep	Mon, 1 Oct
4	2	Game A.I.	Mon, 1 Oct	Mon, 15 Oct
5	1	Text Based Games	Mon, 15 Oct	Mon, 22 Oct

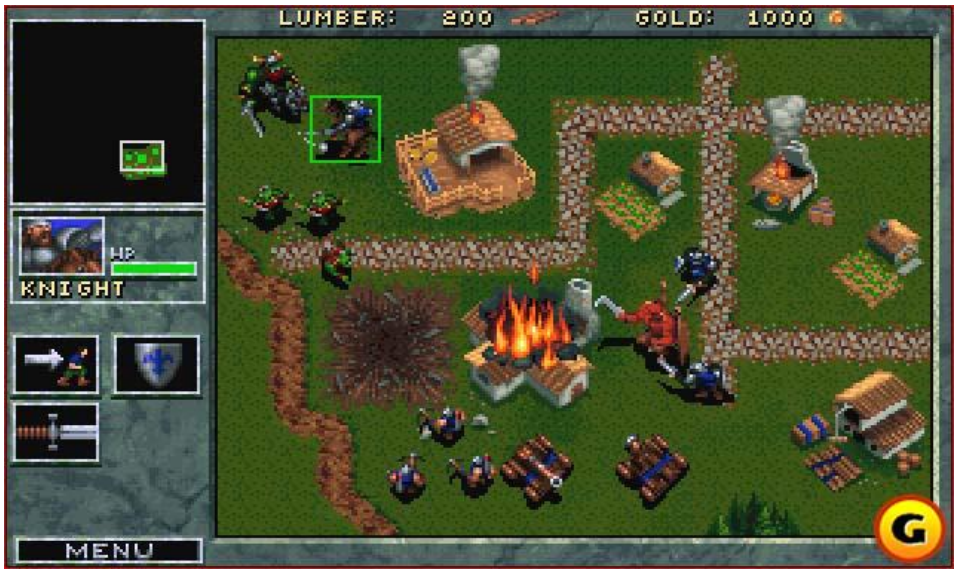

Figure 2: Warcraft 1

1. Game Design & Technical Specification Document

Before you get started with the actual coding part of your game, you will need to design and outline your game so that you have a clear picture of what you have to do. You will not have covered sufficient material in class in order to outline your plans for the collision detection or A.I. sections, you can keep these brief.

In this document you will need to cover a few sections in which you explain your plans for the game, along with the features you want to implement. This is not a formal Game Design Document, but you are required to cover non-technical aspects briefly. The aim of this assignment is to get you to plan ahead and know what you will need to do over the rest of the semester. The sections that you need to include in this document are listed together with a brief description. Please read Game Review examples to get a better feel of what must be included.

- ### Title Page
- ### Executive Summary
  This is a summary of the game you will be making, this will tell the reader exactly what your game is and is usually just a simple paragraph.
- ### Style & Theme
  Describe the theme you have chosen for your game.
- ### Story
  This section is optional if you are planning on including a story in your game.
- ### Collision detection
- You will not have covered sufficient material in class to provide a proper description for this section. All that is required is for you to describe what kind of collision detection and collision response your game will need, must it be highly accurate, fast, or any other requirements for your chosen game. This section can be left brief.
- ### Game Features
  Here you will list the different features you are planning to implement. Explain each in detail and how you will implement them into your game. Consult the Game Features practical for information on what you need to discuss here.
- ### Artificial Intelligence
  You will cover A.I. in lectures later on in the semester and as such you need to just describe the type of A.I. you need for your game. What will their primary task be? This section can be left brief.

This does not need to be the final description of your game and you do not need to stringently stick to it during development. However, it is expected of you to at least attempt to stick with your original design (even if only partially). Things such as only implementing part of your planned features or changing your genre or style will not warrant any penalty.

If you are unclear on whether you will have the time to implement certain features, it is best to include them in this document and drop them from the final implementation if need be.

Marking

You will be marked based on document structure, sensible planning, and clarity of communication. Your overall game design will be evaluated and feedback will be provided on the feasibility and level of difficulty of your chosen design. While there are no formal length requirements for this document, it should be as long as you need to describe your game but try not to waffle. It is recommended to have around 3 pages (800-1200 words). Please **submit a PDF file** of your work as this guarantees document compatibility.

Submission date

You have one week to complete this practical, hand in by 10:00am Monday, 27rd August.

2. Collision Detection & Bit-Checking

The basic game which you have been given has basic and inefficient bounding-box collision detection system which is $O(N^2)$. This is very inaccurate for non-rectangular sprites. You will need to implement a pixel-checking function to perform accurate testing. This is the bulk of the assignment and will earn you up to 65% of the marks.

To get the further 35% you will need to improve the efficiency of the collision detection system. This can be done by implementing an entirely new collision detection system or through clever use of data structures to eliminate unnecessary checks. You will receive higher marks for implementing better systems.

There is a bonus 15% in this practical for improving the bounding-box test with a more suitable bounding-shape test. This should only be attempted after the above two sections have been completed. A simple circle bounding box test is not that impressive.

Marking

You are required to attempt the efficiency increase section of this practical, if there is no evidence of attempts at improvements you will be penalised 20%. There is a possible mark of 115% in this practical; however, it will be capped at 100%.

Section	Marks
Bit-Checking	65%
Advanced Collision System	35%
Bounding Shape	15%
Total	**115%**

Submission date

You have two weeks to complete this practical, hand in by 17:00pm Monday, 10th September.

Important Note

After this practical you will be provided with a model answer for the bit-checking function and are free to use the $O(N^2)$ system in the event your implementation does not work. You are encouraged to keep your own systems.

3. Game Features Implementation

The main goal of this practical is to implement a set of features into your game. You will be required to include three different features in your game. Marks are awarded based on the level of difficulty to implement the features, and as such three moderately difficult features will be needed. Try not to include too many "light" features as this can clutter the game and reduce playability.

Below is a list of some possible features, you are not restricted to this list only. If you have any queries about possible features, feel free to contact the TA for assistance.

Possible features:

- Winning conditions or scoring system
- Health packs and power ups
- Extra weapons
- Different types of enemies
- Character Development (RPG style)
- Menu
- Saving/Loading
- Sound
- Multiple Stages
- Multiple Difficulty Levels
- Menu Screen
- Random Map Generation
- Creative Visual Effects

In the game provided to you there is a basic grid based system, you can choose to leave it as is or improve upon it to get higher marks. This grid system will primarily be used in the A.I. section for the path finding algorithm. You will need to include some form of walls or obstacles in your game such that the A.I. can use them to navigate around to demonstrate the path finding in the next practical. There will be marks awarded for level design and style so pay attention to this when designing the levels.

In preparation for the final practical in which you add A.I. to your game, you will need to incorporate the very basics of A.I. in this practical. Simple A.I. units that move directly to the player's position and shoot or something will earn you the full 10 marks. Read the A.I. practical so that your basic implementation you do here can be used in the next practical. **Failure to include even the most basic form, or attempt, to A.I. will result in harsh penalties.**

Marking

You get up to 60% of the mark for implementing three different features in your game, please remember to **document ALL** features in your readme file to prevent the tutors from missing a feature when marking.

For not modifying the current grid design for the game you will get only 3/10 for the environment setup. Improving upon it will get you up to 6/10. However, breaking it or removing it will result in 0/10. The remaining marks are awarded for environment style and setup in the game (how good it looks).

The A.I. system is integral for work on the next practical and thus you need to have just the very basics of A.I. in place. For this you will get up to 10% but should you not include this a penalty of 15% will be applied along with getting 0/10 for this section.

You will receive up to 20% based on the playability and how enjoyable your game has been. This mark will be calculated on the average mark given by all of the tutors to prevent any biasing.

Section	Marks
Game Features	60%
Environment Setup	10%
Basic A.I. Setup	10%
Playability & Enjoyment	20%
Total	**100%**

Submission date

You have three weeks to complete this practical, hand in by 10:00am Monday, 01 October.

4. Game A.I.

For this practical you are required to implement an A* path finding algorithm for use with your A.I. units. You will need to have objects in your environment that the A.I. can navigate around to demonstrate the algorithm. In order to achieve this you will need to implement a 2D grid based approach and divide up your environment accordingly.

You will also need to implement a method to visually indicate the path the A.I. will be following in order to get to the player. This can be done by drawing lines along the calculated route.

The second part of this practical is to implement a more advanced A.I. system. Examples would be that of grouping behaviours (swarming) or finite state machines, or any other more advanced A.I. functionality. You could even have the A.I. target explosive devices situated near the player should your game feature such devices.

Marking

By completing the A* path finding algorithm to allow your A.I. to move around the environment you will receive up to 80% of the total mark. The final 20% of the marks come from the implementation of the more advanced A.I. system.

Section	Marks
A* Path finding	80%
Advanced A.I. System	20%
Total	**100%**

Submission date

You have two weeks to complete this practical, hand in by 10:00am Monday, 15th October.