Team Software Engineering Group 37

Abstract

[Draw your reader in with an engaging abstract. It is typically a short summary of the document.   
When you’re ready to add your content, just click here and start typing.]

Project Final Report and Artefact

Creature Game

Contents

[Software Engineering 3](#_Toc32427928)

[Agile Project Flow 4](#_Toc32427929)

[Version Control 5](#_Toc32427930)

[Implementation 6](#_Toc32427931)

[Story 6](#_Toc32427932)

[Basic Game Flow - Overview 6](#_Toc32427933)

[Basic Game Flow By Section – Breakdown 8](#_Toc32427934)

[Instantiation and Class / Object Properties 8](#_Toc32427935)

[Object List 8](#_Toc32427936)

[Object Breakdown 8](#_Toc32427937)

[Interface For Creature 9](#_Toc32427938)

[Interface For World / Environment 9](#_Toc32427939)

[Interface For Sound 10](#_Toc32427940)

[Interface For Target 10](#_Toc32427941)

[Interface For Position 10](#_Toc32427942)

[Intro / Game Start Animation 10](#_Toc32427943)

[Main Program Loop 11](#_Toc32427944)

[Influences 12](#_Toc32427945)

[Testing Strategy 13](#_Toc32427946)

[Whitebox 13](#_Toc32427947)

[Blackbox 13](#_Toc32427948)

[Test Results 13](#_Toc32427949)

[Release 14](#_Toc32427950)

[Evaluation 15](#_Toc32427951)

[Group Work Conclusion 16](#_Toc32427952)

[Rectifying Deficiencies 16](#_Toc32427953)

[Contribution Of Each Group Member 16](#_Toc32427954)

[Evaluation of Each Group Member 17](#_Toc32427955)

[Development Targets 17](#_Toc32427956)

[Artefact and Media Materials 18](#_Toc32427957)

[References 19](#_Toc32427958)

[Image Credits 20](#_Toc32427959)

[Licensing 20](#_Toc32427960)

[Appendix 21](#_Toc32427961)

[Design Sketches and Ideas 21](#_Toc32427962)

[Figure 2 Game Opening Idea 21](#_Toc32427963)

# Software Engineering

800 words

The strategy that the group has followed falls within the Agile methodology which comprises various approaches to software development in which “*requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s)*” [1].

This description fits perfectly with a group that has been randomly put together to create a software artefact for a targeted audience. The group members are comprised of three Computer Games and three level 2 Computer Science students.

The software artefact that the group is producing is a Serious Educational Game (SEG), also known as a serious game or applied game [2] for school children. The project aims to teach children aspects of science, biology and ethical consideration by studying how simulated simple creatures, abiding by simple rules, behave and how this behaviour changes as the environment, food supply and crowding conditions change. The player can place food and give the creatures a target to aim for while trying to keep one creature alive longer than the others.

The Computer Games students use the Unity Games Engine [3] within their course and therefore this became the focus for our implementation. The Computer Science students have their course centred around scientific methods, algorithms and the Object Orientated Programming paradigm. The strategy adopted needed to be one that not only brought the skills of the group members to bare but would allow each member to work independently, to some degree, and fit within the time management requirements that each member has while studying other modules.

Alongside the Agile development strategy there is an element of DevOps [4] whereby each member also acts as developer, tester and the end user, and can use their own feedback from Blackbox testing and game playing to feed back into the development cycle in a rapid manner. In addition, each group member has a similar requirement to the target audience in that their studies also entail learning about ethics and understanding some of the underlying topics that the Game attempts to teach.

The waterfall model of software development “*is a breakdown of project activities into linear sequential phases, where each phase depends on the deliverables of the previous one”* [5] was considered initially but discounted due to the fact that feedback, from group members as well as other testers, is required to influence game play and features within the game. During the research phase the group listened to the opinions of a head teacher at a local school who pointed out that game play would be crucial to engagement with school children. The Game must be playable to be able to get this feedback from the target audience. The “User Experience” will be key to the success of the product and feedback may alter many aspects of the game feel / play at any stage and therefore a continual iterative cycle of many aspects is more appropriate.

Scrums [6] as Sprints [7] have been employed within the Agile strategy. Once per week the group members met to work on specific features together at the same time, especially where one of the programmers was having trouble implementing an aspect of the game. These “Scrums” could be mini teams of 2 or 3 members in order to push the project forward. As key milestones arrived within the timeline of the Gantt chart [8] Sprints were employed whereby the scrum went through the backlog of game features marked as not done and concentrated on one of these to get it finished.

## A picture containing screenshot Description automatically generatedAgile Project Flow

The project flow is as follows:

1. Project planning is done and sets requirements for the game
2. Features and Ideas are defined
3. Plans, Features and Game Design are documented and updated continually as feedback comes in and changes some features and ideas
4. Programmers refer to documentation and code based upon features
5. Programmers do their own testing and bug fixing. As they test, they might find features need improving or adding and ideas may fail, or new ideas are realised
6. When required scrums are setup to overcome problems. They test and bug fix. Scrums are employed as an when necessary
7. As milestones arrive Sprints are employed to finish work on time.
8. Formal Black and Whitebox testing are carried out continually and feedback may influence features and ideas of game play
9. Evaluations of beta / test releases, Release candidates all can influence features and documentation
10. Final release – Feedback still sought which could influence features and find bugs

## Version Control

Versioning is controlled directly within Unity [9]

# Implementation

2000 words

## Story

Creatures move independently from each other. There can be many creatures. These creatures need to move, eat, grow older, grow weaker / stronger, breed and interact with each other and other objects such as walls or obstacles. Each creature has a look / graphics which can be scaled and animated. Each object also has a look / graphics which can be scaled and animated. Interactions trigger sounds. There is a soundtrack / music playing in the background. The player has a score. Metrics are recorded as to what and how often something happens, such as how many generations of creatures have been produced, how far each creature has moved, how much food was consumed. The game can be saved or restarted from a previous saved state. The game can be ended early. The whole environment can be zoomed / scaled.

## Basic Game Flow - Overview

|  |  |
| --- | --- |
| A close up of a sign  Description automatically generated | Creature class  World / Environment class  Sound class  Configuration / scores class  Intro opens with an animation placing the game in the context of a microscope  The main game play screen display is shown, all creatures, food, and targets shown.  User Interface symbols and game state are show  User interacts with creatures / sound effects  This section loops until the program exits  Outcome is displayed, animation returns view back to original table / microscope |

## Basic Game Flow by Section – Breakdown

## Instantiation and Class / Object Properties

The game is to be broken down into modules / objects. Each object is a standalone class utilising Object Orientated Programming. Each object can be written independently by different people. Each object will have an interface allowing other objects to interact and access properties and methods within the object.

|  |  |
| --- | --- |
| Object List | |
| 1. Creature Class 2. Food Class 3. Walls / Blocks / Object class 4. Sound class 5. Metric class 6. Score / Game state class 7. World – Environment class 8. Animation class 9. Movement class 10. Target Class | 1. Position class 2. WorldCells Class 3. Menu class |

|  |  |
| --- | --- |
| Object Breakdown | |
| 1. Creature class    1. Has an age (integer)    2. Has a speed (Float)    3. Has a position (x,y) (Integers)    4. Has a health level (Integer)    5. Has a size (Float)    6. Has a hunger level (Integer)    7. Has a method of moving (Function)    8. Obeys simple rules (Imposed)    9. Can trigger sounds (calls function) (raycast or collision)    10. Has a look / style / image (png / vector)    11. Has an animation state (Integer)    12. Can be deleted | 1. **Food class**    1. Has a size (Integer)    2. Has a position (x,y) (Integer)    3. Has an energy value that gives health to creatures (Float)    4. Plays sound when being eaten (Function)    5. Has an animation state (Integer)    6. Has a look / style / image (png / vector)    7. Can be deleted |
| 1. Wall / Object class    1. Has a position (x,y) (Integer)    2. Has a size (Float)    3. Has a look / image (png / vector)    4. Triggers sound when interacted with    5. Can be placed (By User)) | 1. **Sound class**    1. Contains sounds for game    2. Can play / stop    3. Can load a sound |
| 1. Metric class    1. Number of creatures (Integer)    2. Min creature age (Integer)    3. Max creature age (integer)    4. Average creature age (Float)    5. Number of generations (Integer)    6. Average health (Float)    7. Average speed of creature (Float) | 1. **Player score class**    1. Overall score (Integer)    2. Number of interactions (Integer)    3. Time survived (Integer for seconds)    4. Amount of food used (Float) |
| 1. World / Environment class    1. Number of cells (Integer)    2. X size (Integer)    3. Y size (Integer)    4. Background colour (Integer)    5. Background image (png / vector)    6. Scale (Float)    7. Has a name (String) | 1. **Animation Class**    1. Creature dies    2. Creature is born    3. Food shakes |
| 1. Movement class    1. Basic move toward target    2. Simple Dijkstra movement    3. A\* Movement    4. Random    5. AI Movement | 1. **Target class**    1. Position (x,y) (Vector)    2. Image / colour (Integer)    3. Automatic / User place (bool) |
| 1. Position    1. X (integer)    2. Y (Integer) | 1. **World Cells**    1. Stores object types (integer) |
| 1. Menu Class    1. Can Exit    2. Can turn sound off / on (bool)    3. Can display “About” page    4. Can change volume (Integer)    5. Can start game    6. Automatic target (bool) |  |

|  |
| --- |
| Interface For Creature |
| 1. Creature::Creature(Position, Age, Health, Speed, Movement\_Style, Colour) 2. Creature::getPosition() returns (x,y) 3. Creature::setPosition(Position) – expects Vector (x,y) 4. Creature::setColour(Colour) – expects Integer 5. Creature::setImage(Image\_Number) – expects Integer 6. Creature::addHealth(Float) 7. Creature::reduceHealth(Float) 8. Creature::getAge() – returns Integer in days 9. Creature::draw() 10. Creature::playSound(Sound\_file) – expects Integer index of sound |

|  |
| --- |
| Interface For World / Environment |
| 1. World::World(Size, Colour, Background\_Image, Name\_str) 2. World::draw() 3. World::setMusic(File) 4. World::getXSize() 5. World::getYSize(); 6. World::setScale(Float) 7. World::Cells[] |

|  |
| --- |
| Interface For Sound |
| 1. Sound::Sound(Array of sound files) 2. Sound::play(Integer, bool) – play this sound file, loop 3. Sound::stopAll() 4. Sound::stop(Index of sound file) |

|  |
| --- |
| Interface For Target |
| 1. Target::Target(Position) 2. Target::getPosition() 3. Target::setPosition(Position) |

|  |
| --- |
| Interface For Position |
| 1. Position::Position(X, Y) 2. Position::getX() 3. Position::setX(X) 4. Position::getY() 5. Position::setY(Y) |

## Intro / Game Start Animation

The game will open with a short animation showing a table and a microscope on top which sets the scene for the player. The animation slowly zooms in upon the microscope, and specifically, the Petrie dish. This is achieved using 3D SVG images*, [1] Microscope Low-Poly* created by Iwan 'qubodup' Gabovitch and used under a Creative Commons License, [*2] XYZ Table,* (licenses referenced inside program code and shown in program credits menu), and rendered using Unity’s standard Vector Graphics package version 1.0.0. The camera path is using the BezierContour class provided by Unity.

## Main Program Loop

The main program loop is of a standard traditional game engine format and detailed at <https://docs.unity3d.com/Manual/ExecutionOrder.html> but the simplified version is:

|  |  |
| --- | --- |
| A close up of a sign  Description automatically generated | The game is started / Run by the user  Class definitions / Object instantiation  Files loaded / sound / images  **LOOP START :** This game doesn’t use physics per se but potential improvements and updates to the game may do  Mouse input events read  Keyboard input events read  Creature positions updated, as is health, scores, metrics and any other changes and updates to variables and data either by automatic changes or triggered via input events  Creatures, Food, Walls and the World are drawn to the screen. The loop may pause at this stage to synchronise with any specific timings such as frame rate.  **LOOP END -> LOOP START**  Objects deleted / Game ends. Includes saving of metrics data, game state. |

## Influences

Breaking down the main loop all the inputs change either or all the food placings, the target and each creature. These inputs influence the behaviour of the creature. The inputs are used within the game’s logic section to make decisions as to the exact nature of the influence. The creatures seek out the target, but time affects whether they need food. The creatures will either head toward the target or seek out food. Time also effects the creatures age. The creatures age and hunger effects the speed of movement. Another influence on movement is the position of other creatures and whether they block each other. All these inputs effect how the simple rules the creatures obey are observed. The positions and state of all objects are updated within the game logic section before being rendered to screen.

A screenshot of a cell phone

Description automatically generated

# Testing Strategy

1500 words

Testing was continually carried out by all members as code was being written in respect of code review and discussions. Both minor and major bugs get caught during this informal testing stage and although this is part of normal code writing further testing was formalised.

A formal test procedure has been written for both Whitebox and Blackbox testing as follows:

## Whitebox

All code is contained within modules / objects as per the Object Orientated Programming paradigm which means each object can be tested separately. The specifications of each object, and the functions / methods within, and their interfaces have been documented within the **Instantiation and Class / Object Properties** section of this report. Testing is done both within the overall game by changing parameters but also by placing each object within a test environment. The test environment can be either a separate program written specifically to test the object but also within a separate protype of the game. A separate prototype game has been written to test algorithms and to see that their behaviour and operation is appropriate logically. Since the game is 3D in nature the proper workings of an algorithm are better seen within the 2D prototype version.

The test procedure and results can be seen within the appendix of this report.

## Blackbox

Testing is to be carried out by all group members as well as random subjects. Subjects include the target audience of school children and those who are likely to be delivering and using the game for educational purposes.

Testing included User Interface, User Experience, game play, the game feel, and whether it complied with out definition of how the game operates.

Using random subjects and having them fill out a form requires ethical consideration with regards to the data that is captured and how this data is used. The test data that is collected will be made anonymous and will be used solely for the purpose of creating and improving the game during an iterative type process. No names or details about the tester will be collected other than age. The tester’s age is required in order to put the results into context. When reviewing the data, the age is all that is required to determine whether the results come from someone expected to deliver the game as a teaching as or as a user.

Data will be stored, in an anonymous form and kept for as long as it takes to mark our project. Once marked this data will be deleted. The data requires storage as it is evidence that Blackbox testing was carried out as part of the project.

A copy of the Blackbox test form can be seen within the Appendix section of this document along with data.

## Test Results

Results were fed back into team meetings, discussed as necessary with missing features and bugs rectified and or documented as to what steps were needed to address any issues found.

# Release

In this section you should describe how your released your artefact. Specifically, you should describe

what platform you released it on and why that was an appropriate venue. Furthermore, you should

describe what precautions you put in place to manage issues-on-release.

Since the “Creature Game” game has only reached a pre-release beta version, it has been released within Github. This provides the ability, not only for group members, but also the wider community to take up the project and progress to the next stage.

When it is finished it is envisaged that it would be released within the Microsoft App Store since the target audience is school children and teaching professionals who predominantly use MS Windows operating system. This target audience require easy installation and automatic bug fixes and updates on a known trusted platform that they are fully familiar with.

In keeping with general Open Source principles, the source code and all related files will also be available on Githib under a GPL 3.0 license

# Evaluation

Once your artefact is released you should conduct a detailed evaluation. The evaluation will be partly from you, but should also include insights from users (reviews for example). Your evaluation should include a discussion about how you could develop the artefact further in the future. 1500 words

# Group Work Conclusion

In this section we will expect a co-authored reflection on how your group has worked together throughout the project. You should include some discussion about your successes, as well as areas that you could improve. This should be an honest and critical statement and should help you to set personal development targets for future work. As part of the reflection you should include a table which details each member’s individual involvement in the work (as a percentage). 800 words.

## Rectifying Deficiencies

The interim report, and it’s marking, published in November 2019 highlighted several deficiencies in the group’s approach to this project.

These deficiencies have been addressed, with the main issue being the fact that group member roles were not clearly defined. From the start of the implementation of the game, roles were defined as follows:

|  |  |
| --- | --- |
| Group Member | Main Roles |
| Andy Perrett (18684092) | Testing, Documentation |
| Morgan J Onley (15620639) | Graphic Design, MDA |
| Daniel Monk (18680710) | Types of Fun, Supervisor liaison |
| Shane Lawson (18680054) | Game Play |
| Dylan Ryal (18673821) | Game Feel |
| Joshua N Towell (17664923) | Game Design, Marketing |

Each main role has sub roles or tasks which have been delegated in some cases to other group members. The group member takes responsibility for their role. The Gantt chart has been updated to reflect these defined roles and it was seen that the original Gantt chart was too optimistic, as well as unrealistic, with time scales so these time scales were also updated.

## Contribution Of Each Group Member

|  |  |
| --- | --- |
| Group Member | Contribution Percentage |
| Andy Perrett (18684092) | 16.6667 |
| Morgan J Onley (15620639) | 16.6667 |
| Daniel Monk (18680710) | 16.6667 |
| Shane Lawson (18680054) | 16.6667 |
| Dylan Ryal (18673821) | 16.6667 |
| Joshua N Towell (17664923) | 16.6667 |

Establishing contribution metrics in order to produce, (and reduce), a member’s contribution down to a single percentage number was extremely difficult. The group decided the best approach was to take a subjective look at each member’s efforts. A lot of contribution comes from reading, thinking and talking which can be assessed by simply seeing how engaged a member is. Other contributions such as documentation, code writing and graphics can be seen in what has been physically produced and comparing one form of contribution to another, and the importance of each, can only be subjective.

Since each member attended meetings, gave opinions, discussed code, produced graphics and sketches, helped the organisation and actively engaged in the project, as well as put in the hours to achieve the overall project, the group decided the simplest approach was to ask a question. The question being, “are we all happy with each other’s level of contribution?”, the answer was yes.

The contribution levels were therefore determined to be equal. This has the advantage of avoiding conflict when discussing who did more and who is more important to the project. Each group member was equally as important as any other in respect of completing the project.

## Evaluation of Each Group Member

Evaluation of each group member was continual, for example, during initial game coding it became apparent that both Dan and Morgan had started, independently, work on producing the initial introduction sequence and graphics. This was quickly rectified and the reason for the failure determined to be a lack of communication and not using collaboration within Unity. Another example highlights the need for inter-role communication in which Andy was not aware of how Unity represented Class objects and their properties in comparison to how this is done within plain C# / C++, which again was quickly rectified by Dan demonstrating how Unity is used to build games and sharing his documents with all members.

Lack of communication skills by all group members was determined by this continual evaluation process and rectified with more meetings, both all together and utilising pairing up “get togethers”. The sharing of code and documents, via Github, MS Teams software, Discord chat software, and the use of Trello helped greatly. The Gantt chart was made interactive and each member had access to the chart allowing them to update progress.

Guys, entries in this section aren’t criticisms per se, we simply must be critical and highlight weaknesses and strengths for each person

## Development Targets

Each group member needs to work on their communication skills. The examples given previously highlighted this. The target for each member is therefore to learn to identify examples of problems caused by a lack of communication and simply share what they are doing, share their progress and discuss, document what they have done, and plan with others more often. A general question was thought up by the group during a meeting to help with this issue. It was decided that we should all ask the question “do the other group members understand what I am doing and why, do I understand their roles and why?” at regular intervals.

Specific individual developmental targets have been set as follows:

|  |  |
| --- | --- |
| Group Member | Development Target |
| Andy Perrett (18684092) |  |
| Morgan J Onley (15620639) |  |
| Daniel Monk (18680710) |  |
| Shane Lawson (18680054) |  |
| Dylan Ryal (18673821) |  |
| Joshua N Towell (17664923) |  |

# Artefact and Media Materials

In your report you will need to provide details of how your artefact can be accessed for marking. This should be in the form of a link to the platform that the final project has been released on. If your platform only hosts the executable, you should also provide a link to where the code can be accessed (GitHub for example). In addition, you should provide a link to some ‘media materials’ that could be used to promote your artefact to its target audience. This should include a video no longer than 2 mins long.

# References

[1] <https://en.wikipedia.org/wiki/Agile_software_development#Agile_software_development_practices>

[2] <https://en.wikipedia.org/wiki/Serious_game>

[3] <https://unity.com/>

[4] <https://www.atlassian.com/devops>

[5] <https://en.wikipedia.org/wiki/Waterfall_model>

[6] <https://stackify.com/what-is-scrum/>

[7] <https://www.atlassian.com/agile/scrum/sprints>

[8] <https://www.projectmanager.com/gantt-chart>

[9] <https://docs.unity3d.com/Manual/Versioncontrolintegration.html>

# Image Credits

[1] “Microscope” is from : <https://opengameart.org/content/microscope-blend-low-poly>

The License is <https://creativecommons.org/licenses/by/3.0/>

Attribution Instructions: Microscope Low-Poly by Iwan 'qubodup' Gabovitch <qubodup@gmail.com> licensed under CC-BY-SAv3

[2] XYZ Table

# Licensing

Game distribution info <https://answers.unity.com/questions/852628/i-finish-a-game-and-i-want-distribute-it-what-i-ne.html>

# Appendix

## Design Sketches and Ideas

A picture containing indoor, text

Description automatically generated

### Figure 2 Game Opening Idea