G51FSE Development Portfolio

***# For our benefit --------------------------------------------------------------------------------------------------------------***

***Expected Content:***

* *Game concept, requirements and specification (functional and non-functional requirements)*
* *Design of your game including any rules, controller behaviour, interaction design, scoring mechanism and game overview*
* *Evaluation of prototype and design refinements*
* *Implementation notes and description of the development methodology*
* *Evidence of testing and debugging including a write up of test cases and /or use of a test framework*
* *Evaluation of the game through user evaluation*

***Advice:***

* *Diagrams are a good way of expressing design ideas, there are many formal varieties but even simple block diagrams can help get your point across clearly.*

***Coursework Objectives (tick when done):***

* *To perform the process of software engineering from start to completion, demonstrating evidence of ability and aptitude for each stage of the process.*
* *To be able to program and develop effectively in pairs.*
* *To demonstrate original thought and creativity in software production.*
* *To show use of the appropriate software engineering methodologies and diagrams.*
* *To comply with the attendance procedures specified for the lab sessions.*
* *To gain experience in working with external code and libraries*
* *To apply programming knowledge to a new language*
* *To be able to manage source code through version control systems *

***# For our benefit --------------------------------------------------------------------------------------------------------------***

Introduction

Our assignment was to create a two dimensional game using Python and Pygame for a small games development company NG2 Games. The company’s focus is not only on imaginative and inventive games but quick software delivery. We were to develop the game from an initial specification, onto design, prototyping, refinement, implementation, testing and finally evaluation.

*As our employer was interested in quick software development as stated in the specification, we both agreed that rapid prototyping would be required to complete the task on schedule. Our chosen Software Engineering Methodology was therefore agile methods/eXtreme Programming.*

REST OF IT WILL BE WRITTEN IN FIRST PERSON (AS IF WE ARE FILLING IT IN AS WE ARE GOING ALONG)

Initial Specification

WE SHOULD FOCUS MORE ON GENERAL SPECIFCATIONS HERE THAT ALL GAMES MUST ADHERE TO AND THEN AS WE START TALKING ABOUT PROTOTYPES WE CAN MENTION GAME RULES ETC. AGILE REQUIREMENTS ARE ELLICTED MORE IN THE USER STORIES STAGES.

Functional Requirements (services or functions):

* Single player game
* Users will interact with the game using the keyboard/mouse
* High scores will be recorded
* **ADD MORE**

Non-Functional Requirements (constraints on system or development process):

* Employer is interested in rapid software delivery
* Must be developed using Python and Pygame
* Must satisfy a range of audiences in order to be profitably viable
* Must be sensitive to religion/race/gender etc.
* Game rules must be made available to users
* **ADD MORE**

User Stories (very important to agile as features will be elicited from this)

* As a first time player, I will need to be able to see the rules of the game
* As a dedicated player, I would like to see mine and other people’s high scores to check out the competition
* As a generic player, my skills we have to be tested in order to keep me interested in playing the game and to prolong my enjoyment
* As a stakeholder, the game must be profitable for my company
* **ADD MORE**

Prototyping

Overview

After formalising our initial specification into user stories, we have agreed on developing an interactive projectile game “Angry Joolz”. Users will launch characters resembling our module Lecturer at various objects. The characters, obeying the laws of physics, will destroy the targets if they are hit. Points will be awarded on the level of destruction, with the targets getting increasingly harder to hit as levels progress.

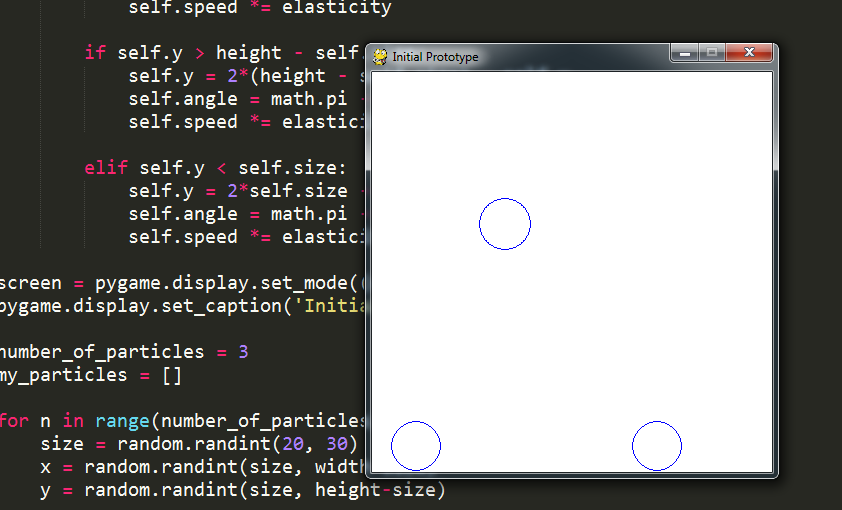
Features of the Game elicited from User Stories

* A user will interact with the game using the mouse
* Clicking and dragging the characters backwards increases the projectile force
* The angle of the projectile can be set with the mouse
* Points will be awarded for the level of the targets destruction
* Targets will become increasingly difficult to hit as the levels progress
* If a user fails to hit a target within a certain number of lives, the game is over
* **ADD MORE/NEED TO START SIMPLY AND BUILD UPWARDS**

Game User Stories?

* As a [blah] I will [blah] because [blah]

Initial Throw-Away Prototype

****

*Figure 1: Initial Throw-Away Prototype*

Having followed several tutorials online [1][2] (**INSERT REFS 1,2**) we have decided that in order to adhere to the assignments’ specifications we cannot continue with our initial game “Angry Joolz”. The prototypes are far too time consuming to produce and we do not have the time constraints to implement the functionalities necessary to make the game both function correctly and be enjoyable to all users. Some of the code we have experimented with in our initial prototype is on hand for reuse in future prototypes as we have discussed the possibility of moving obstacles in most of our initial ideas.

Secondary Prototype

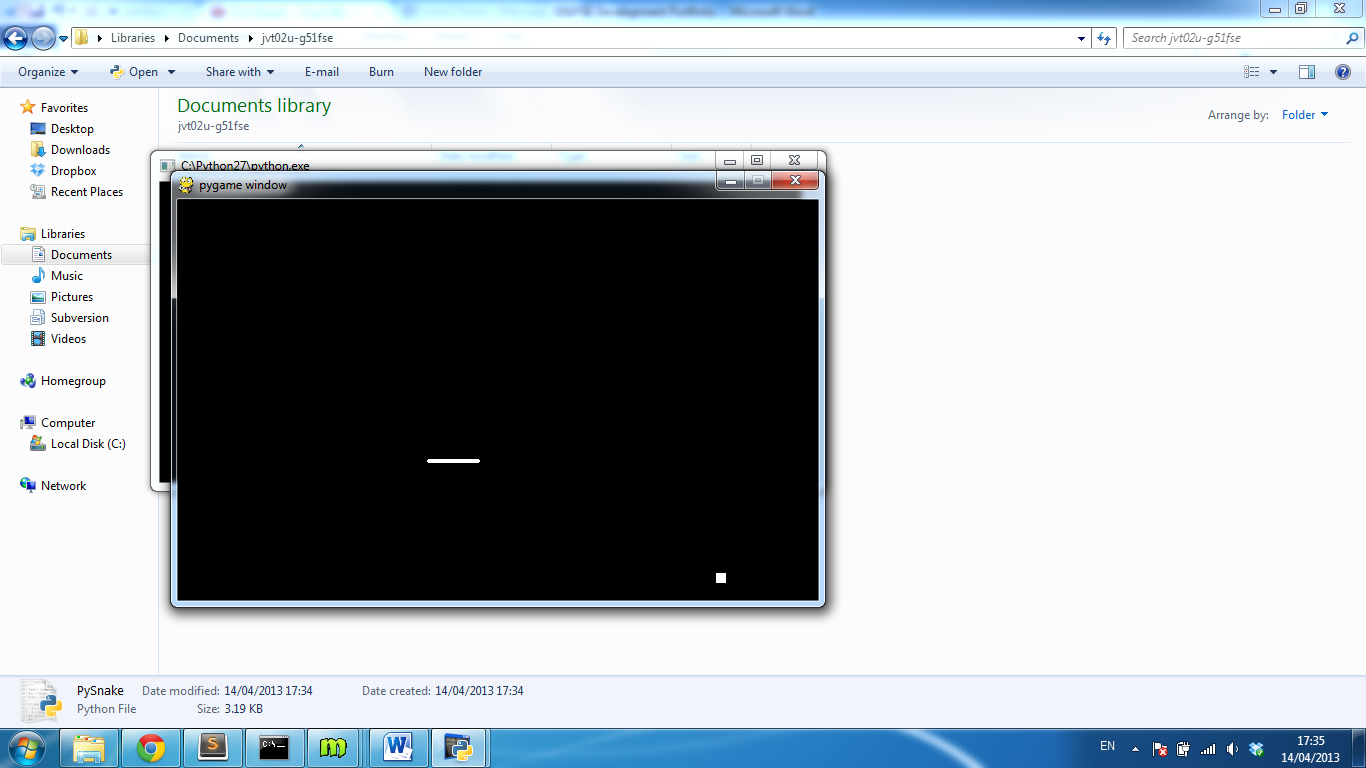
Having run into countless problems throughout the prototyping phases of our initial idea, we have decided to focus on a game much more familiar to both of us. Our new idea extends on the classic 90’s Nokia game, ‘Snake’. We are aware thatour employer insists our game must be inventive, imaginative and profitably viable (therefore not a simple replica of an existing products). ‘PySnake’ will incorporate many new features never before associated with the classic alternative but at the same time reminds users of a familiar, enjoyable game, from years gone by.

Features of the Game /Revised specification still adhering to initial User Stories

* A user will interact with the game using the keyboard
* A user will direct a virtual snake around an arena with the directional arrows
* **THE REST SHOULD BE ON OUR PROTOTYPE SPEC (ADD MORE)**

Game User Stories?

* As a [blah] I will [blah] because [blah]



*Figure 2: Initial Throw-Away Prototype (Secondary Idea)*

[Submitted our initial prototype]

Feedback from Secondary Prototype

* **Can’t find my feedback from prototype**
* **List main points here**

From the stakeholder’s feedback we are now aware that we have to refactor our game code to primarily incorporate the use of sprites. Another key point raised was the fact that a lot of other developers produced a similar game plan. We are confident that our game, when finished, will not resemble any of the other groups’ projects and will continue without a complete redesign.

As we are committed to our choice of game, we will continue developing our secondary prototype, improving it and adding in the remaining features from our specification as to not halt our progress. At the same time we will produce a new prototype away from the main game, which incorporates sprites. As this is a new prototype we can quickly test additional features that have come up in discussions recently (post-secondary specification) without having to worry about unforeseen conflicts arising in large amounts of code. If we agree a feature is desirable, we can then work out how to incorporate this into our main game. The final step would then be to adapt the main prototype to incorporate sprites using the framework provided by the spin-off prototype.

**Something like that to sound ‘agile-y’**

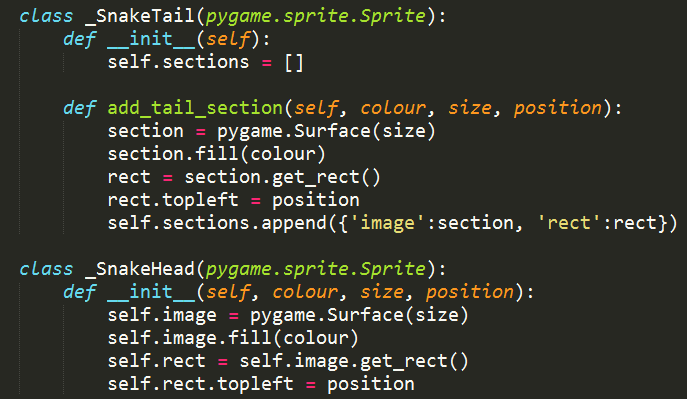
**I TIHNK HERE WE NEED TO REVIEW OUR CURRENT PROTOTYPE AGAINST OUR USER STORIES/REQ SPEC. AGILE PROJECTS STILL HAVE TO BE GUIDED BY REQUIREMENTS**

**Prototype B**

We programmed this prototype to be very basic, concentrating on the use of sprites as was criticized in our stakeholder feedback. During development, we also agreed that we would address another criticism we received from the stakeholders; modularization.

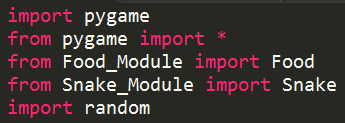
Refactoring

* Separated the head and tail of the snake into two individual ‘objects’
* Introduced basic sprites
* Sprites for the snake are rectangle ‘sections’ that can be filled with any RGB colour
* The snake object, food object and main game code have been separated into modules
* Simple sprite movement diagram below (for the snake)
* ANY OTHERS HERE



*Figure 3: Evolution of the snake object*

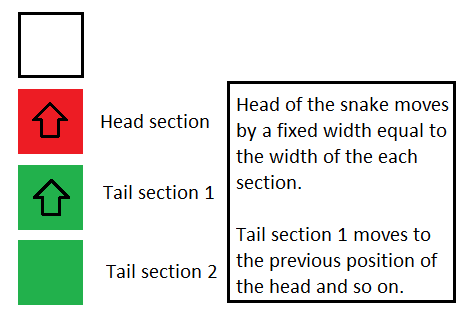
Splitting the snake into two separate objects; head and tail, allows us to use the function **add\_tail\_section** to increase the length of the snakes’ body independently from the head. The snake’s body is initialized by creating a single head followed by X number of tail sections. The number of initial tail sections is variable and could be implemented as part of a difficulty setting.



*Figure 4: Modularization*

Now our objects have been modularized we can see the benefits and when we adapt our main game prototype to this framework, we can use isolation and abstraction to weed-out any redundant code in our project.

Sprite Movement Diagram (to redo in AI)



*Figure 5: Simple Sprite Movement*

Other features explored

* Using a loop, any number of sections can be added to the tail of the snake
* Each new section of the tail, when added, can be assigned with a random fill colour (RGB)

We experimented with the idea of adding multiple sections to the tail at once. This could be applicable in game as a result of the player directing the snake over a bad piece of food. As we can control the colour of each section, we could for example, add X red sections onto the snake as a punishment for a player’s mistake. They could remain permanently throughout the game, for a limited time, or be removed as a consequence of another in-game action.

**If there’s anything you want to play around with add it in here when you’ve added it to the prototype B.**

**Third Prototype**

**Talk about improvements made to the main prototype whilst we were ‘doing’ prototype B.**

**E.g. testing the mode where the screen colour flashes and how it’s horrible**

**Refactoring**

* **produce a design for your python game including, sequence diagrams, state charts and class diagrams**

**Now we’ve introduced modularisation into the spin off prototype we can say that we are going to refactor the main code to the same framework,**

**Produce some UML diagrams for the classes when we’ve done them and maybe state diagrams, e.g. food Spawn->Eaten (Visible->Invisible)**

Fourth Prototype

When we’ve got the main code into classes

Refactoring Prototype B

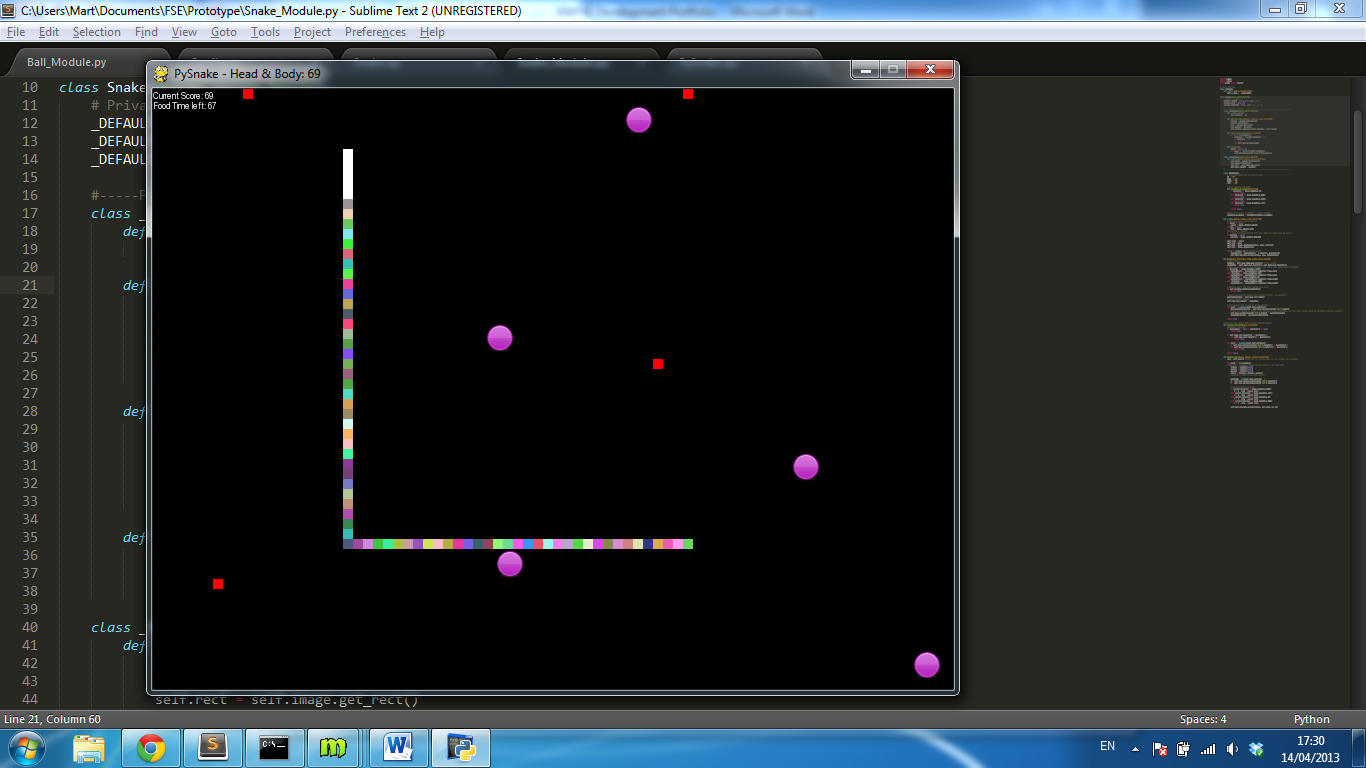
After engaging with the spin-off prototype and utilizing its capability for testing new features rapidly, we have incorporated some additional characteristics which we consider desirable to players. We have also gone a step further with regards to modularization.

* A class ‘Config’ has been created; this holds configuration variables that need to be used in multiple classes.
* In addition to the add\_tail\_section method, we have implemented remove\_tail\_section to remove X amount of tail sections. This will occur as a reward from an in-game stimulus to be determined
* Another method fuck\_tail has been added as a severe punishment. When called, the method will effectively remove the tail section from the players’ sight by changing its colour to black (the same colour as the game arena). The tail will remain active in- game and if the user directs the head of the snake into the tail’s path the game is still over. This punishment is still in development and may be subject to a time limit.

Properties from the main prototype have also been refactored and introduced

* The obstacle balls have been implemented as sprite objects. (Collision detection has not yet been implemented)
* The current score has been displayed on screen (for testing purposes)

As the project currently stands, we have decided to use PNG images for the food and obstacle ball sprites. We will keep to generating surfaces for the snake’s head and each of its tail sections as this gives us access to features such as spawning random coloured sections which we consider an enjoyable and unique feature of the game.



*Figure 6: Refactoring Prototype B*

Possible further enhancements discussed:

* Random sized balls spawned
* Different coloured balls
* ADD MORE

ANYTHING YOU LIKE TO ADD HERE?

HERE WE NEED TO COMPARE OUR PROJECTS FEATURES TO THE USER STORIES AGAIN/REQ SPEC

Fifth Prototype

Refactor main game to include sprites, merge with prototype B

Testing

complete unit tests of your code as you implement, and keep a record of your testing for use in

your design portfolio. *Hint: python has a built in unit testing tool called* unittest. OR use another

standard python test harness

• produce a short evaluation of your work

• evolve your game to include improved playability, graphics and/or multiple levels

Final bit

• complete your python game

• start writing up your Development Portfolio

• perform an evaluation of your game with at least two participants (yes, it can be your Nan).

Appendix:

[1] <http://www.petercollingridge.co.uk/book/export/html/6>