Department of Computer Science

Jamie Birch, Sarah Carter, Ching Chi Chan,

Matthew Ellams, Sam Grant, Alexander Lorimer

**Code** A

**Colony**

\_\_\_REPORT\_\_\_

MS50 Computer Science Conversion,

Software Development,

Group **B**ees

2016

**Declaration of Mark Distributions**

We hereby declare that the following distribution of marks have been agreed by all of the undersigned, and that the work detailed in this document is the sole product of the group members.

Jamie Birch % \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sarah Carter % \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ching Chi Chan % \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Matthew Ellams % \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Sam Sutherland Grant % \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Alex Lorimer % \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Contents**

1. **Introduction**
   1. Background
   2. Aims and Objectives
   3. Target Audience
   4. Definitions
2. **Planning**
   1. Concept Development
   2. Market Research
   3. Initial Concepts
      1. Text Adventure Game
      2. CPU Mario
      3. Shrek
      4. Bees
   4. Final Concept
   5. Competitor Analysis
3. **Project Management**
   1. Overall Objectives
   2. Planning and Scheduling
   3. Gantt Chart
   4. Tools
      1. Pair Programming
      2. Sprinting
      3. Unit testing
      4. Version Control
      5. Continuous Integration
   5. Rescheduling
   6. Evaluation
4. **Design and Implementation**
   1. Initial Design
   2. Risk Assessment
   3. Task Allocation
   4. Final Design
   5. Testing and Debugging
   6. Future Development
5. **Results and Feedback**
   1. Testing Feedback
   2. Results
   3. Evaluation
6. **Individual Evaluation**
7. **Appendix**
   1. Meeting Minutes
   2. References

**Abstract**

Code A Colony is a game in which players are given significant control over a colony of bees. This colony can exhibit complex, collective behaviors determined by player-constructed algorithms. Targeted at Key Stage 3 level students, the aim is to develop and reinforce computing skills that have become a statuary part of the UK curriculum as of September 2014 (with a significant focus on the application of logic and algorithmic problem solving).

We have designed the game to allow players to experiment with custom problem solving/swarm intelligence algorithms, within graphically rich, interactive context. The bees have an intuitive API, and have been endowed with a small amount of memory along with an ability to conditionally interact with both their local environment and with neighboring bees. The controlling algorithm is constructed and manipulated by the player graphically, represented as visual connections between selected nodes (which represent functions) within a graph.

This report details the research, development and management that went into the Code A Colony project, while also documenting our experiences of certain agile software development processes.

**Introduction 1.0**

Technology has changed the way in which we live our everyday lives, from communicating with friends all over the world with a simple click, to the way in which we find a new recipe, or even how we navigate from A – Z. Computers have emerged as ubiquitous devices; increasingly powerful, portable, and intuitively interactive. But this apparent simplicity belies their growing complexity and sophistication. Few of us are capable of seeing past the opacity of modern computers, to reason about their underlying technologies, despite the vast majority of the population relying on digital technology daily.

In the 1980s (at the very cusp of the personal computer revolution) a gap in the computer literacy of the public had been noticed. This provoked the BBC’s Computer Literacy Project and the launch of the BBC Micro (a personal computer designed for computing education). As described by the project editor David Allen, “The aim was to democratise computing. We didn’t want people to be controlled by it, but to control it.”[[1]](#footnote-2) This initiative introduced millions of children to the field of computing, and inspired a generation of computer scientists and engineers.[[2]](#footnote-3)

Yet growing concerns still exist, revolving around a disparity between computing education and the demands of the industry. In 2011 Eric Schmidt (then Chairman at Google) criticised Britain’s education system, stating “your IT curriculum focuses on teaching how to use software, but gives no insight into how it is made.”[[3]](#footnote-4) Inspired by the success of the BBC Micro, and fuelled by the current discourse surrounding computer science education, this report details our attempt at designing and implementing a system to educate and inspire Key Stage 3 level school children about digital technology. The intention is to help demystify core computing concepts to a new audience, reinforcing recent statutory changes to the national curriculum regarding computer science education.

**Computer Science Education 1.1**

In September 2013 the UK Department for Education published a document explaining significant changes to the national curriculum, with a legislative move (at Key Stage 3 level, and earlier) away from general “ICT” and the use of common software, towards teaching fundamental computing concepts such as boolean “logic, algorithms and data representation.”[[4]](#footnote-5) A fundamental aim of this reform is to ensure that all pupils, “can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems.”[[5]](#footnote-6)

Before these changes came into effect in September 2014, concerns about a lack of appropriately skilled teachers prompted millions in investment (from both public and private sources) towards computer science training programmes for secondary school teachers.[[6]](#footnote-7) Never-the-less, in December 2015 the chief of the British Computing Society (appointed by the government to lead these reforms) stated that teachers are still struggling with the new curriculum, and that it could be another five years until it will be successfully taught[[7]](#footnote-8) across the country.

A significant problem appears to be that not all schools are fortunate enough to have access to specialist, appropriately qualified staff. Government investment continues to be aimed at increasing the number of “master” computing teachers, but these targets continue to be missed.[[8]](#footnote-9) Additionally, with roughly 60% of parents not aware of the new curriculum’s introduction in 2014,[[9]](#footnote-10) support for computer science appears to be limited both as school and at home. Additionally the ratio of girls to boys who decide to study computer science at GCSE level is shockingly low.

**Gender Gap 1.5**

Computer science as a subject at GCSE level in 2014 was significantly less popular than ICT. This may have been influenced by initial implementation problems. Nevertheless, the average ratio of girls to boys in computer science cohorts was significantly less than in ICT (see graphs below). Despite this, as part of further reforms, the government announced plans in November 2015 to scrap GCSE and A-Level ICT altogether. The intention is to cast more focus on computer science. As stated in the original document, ministers have “taken the decision not to approve two GCSEs and A levels in a similar qualification space. The IT GCSE and IT A level will not be redeveloped.”

This has sparked significant debate,[[10]](#footnote-11) largely hinging on concerns about further alienation and narrowing of the curriculum for girls.

*Statistics collated Joint Council for Qualifications ‘Results 2015’ http://www.jcq.org.uk/Download/media-centre/news-releases/gcse-press-notice---wales*

*Statistics collated Joint Council for Qualifications ‘Results 2015’ http://www.jcq.org.uk/Download/media-centre/news-releases/gcse-press-notice---wales*

**Industry Stakeholders 1.4**

The recent computing focused curriculum reforms were instigated largely by pressure from tech industry leaders,[[11]](#footnote-12) asserting that significant change will be essential if the UK is to remain globally competitive in certain computer-focused disciplines. Next Gen, for example, is an report published in 2011, strongly recommending reforms the UK curriculum to bring computer science in as an essential discipline.[[12]](#footnote-13) The Next Gen report was responding to declining national competitiveness in the computer game and visual effects industries, along with concerns about increasingly “having to source talent from overseas because of skills shortages at home.”[[13]](#footnote-14)

The report also strongly recommended the appointment of “the best teachers to teach computer science”, and the use of gaming and visual effects within classrooms, suggesting, “Schools harness the ‘cool factor’ and technology potential of video games and visual effects to draw in more young people to computer science.”[[14]](#footnote-15) It is too early to see the industry ramifications of the reforms to the national curriculum, but so far the indications are that the classroom learning experience has not been enough to effectively draw children in, particularly the girls.

**Target Audience 1.3**

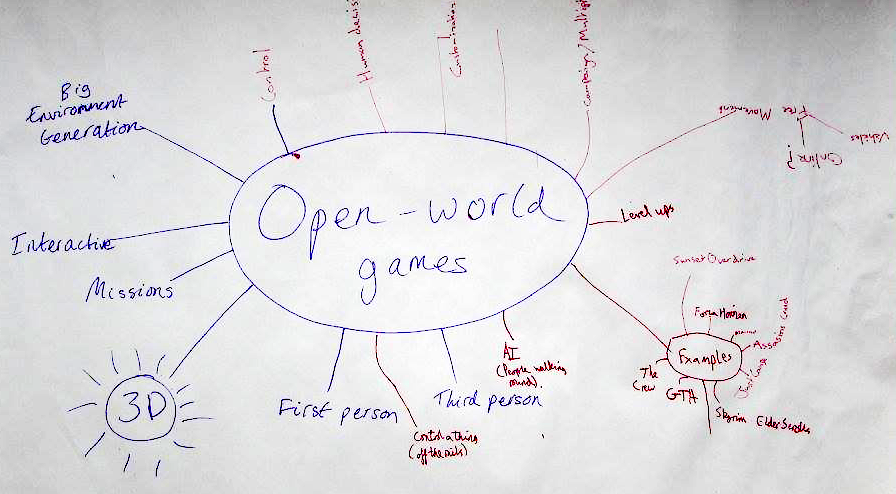
While secondary market research has acted to inform our motivations for this project, our efforts in the development of Code A Colony have been guided significantly by primary market research. We have tried (where possible) to engage directly with Key Stage 3 educators as well as a user sample of our target audience (year 9 students of Bristol Grammar School, aged 13–14, both male and female). This has acted as an invaluable source of information, while allowing us to practice the agile programming concept of short feedback cycles.

**Bristol Grammar School, IT Class Focus Group (Pre-development) 1.3.1**

An initial visit to the school took place on Thursday 8th October, in which we were granted an hour for a focus group with year 9 pupils. For this we devised a series of activities designed to acquire information about IT education, along with opinions and preferences regarding computer games (as well as other topics).

During the first 5 minutes we introduced ourselves and engaged in a class discussion about what the pupils had been doing and learning so far in terms of computer programming. The answers **ranged significantly**, from having little experience at all, to actively engaging in interesting web-scripting projects. We were also introduced to their use of Scratch, **a visual programming language**. This came acted as a great source of inspiration for our project.





We then divided the class into four lines and set up a relay game where the first person in each line ran up to the front of the class to write an interest on an A2 sheet of paper. After several minutes the group with the longest list would win. We then handed out coloured stickers to each pupil and asked them to place a sticker next to the three interests that they prefer (on any of the four sheets of paper). This allowed us to get a ranking of the most common interests, while sparking class discussion.

Next we organised smaller groups, asking each to pick an interest and brainstorm possible connections to the field of computer science. Finally, we asked the groups to design a game relating to this interest, keeping in mind that it must include an educational computer science element. Pupils were encouraged to draw mind maps, and these materials can be found in Appendix 1. The results from these activities are listed below.

Class Discussion & Groups Concepts

* Emphasis on interactive, **explorable**, non-static, **‘open worlds’**
* Connection to **cartoons**, and perhaps recognisable characters
* ‘Mario’ concept , where a player **collects blocks of code** and has to figure out what order they should go in before handing to the CPU
* Importance of **story**-based progression

Pen Relay Game

Highest Ranking Interests

* South Park (4)
* Xbox (4)
* Hockey (3)
* Family Guy (2)
* Football (2)
* FPS CoD / cs:go (2)
* GTA-game (2)
* Open world games (2)
* Rugby (2)
* Antelopes (1)
* Caramel (1)
* Phone (1)
* Shrek (1)
* Social Media (1)
* Strategy (1)

Cartoons

Digital Technology

Sport

Commonalities

In addition to these activities, we also asked pupils to fill out a questionnaire at the end of the class, ranking twelve game aspects in terms of importance to their experience (see pie chart below; ranks on next page).

Game Aspect Questionnaire (out of 10 – lowers scores rank most highly)

| **Category** | **Combined score** |  | **Rank** |
| --- | --- | --- | --- |
| Graphics | 38 |  | 1 |
| Story | 43 |  | 2 |
| Competing with others | 61 |  | 3 |
| Constant interaction | 68 |  | 4 |
| Innovation | 75 |  | 5 |
| Characters | 76 |  | 6 |
| Cooperation with others | 77 |  | 7 |
| Single player gameplay | 91 |  | 8 |
| Checkpoints | 94 |  | 9 |
| Levelling up | 96 |  | 10 |
| Sound design | 105 |  | 11 |
| Replayability | 115 |  | 12 |

The main insights we gained and took forward from these activities were that:

* **Cartoons and comedy** seem to be a useful tool to make game ideas more appealing. Game characters that are caricatured, and themes/challenges that contain an element of humor could be useful devices to draw in our target audience.
* **Story- / stage-based progression** (“levelling up”) was also a very popular concept. This could in some way be tied to an aspect of humour, but also would be very useful in the form of a progressive tutorial.
* **Game environments** that can be **openly navigated**, and freely explored and interacted with are important.
* A sense of **collecting items, points, or “code blocks”** could act as the main challenge, allowing for some element of competition (competing against your previous score, or the score of a friend).

**Bristol Grammar School, meeting with the Head of IT (Pre-development) 1.3.2**

On top of our detailed discussions with school pupils, we were also granted a 30 minute meeting with Amy Finney, the head of IT at Bristol Grammar School. Her very informative insights regarding the development of a computer science education game are summarised by three key points.

The game should be:

* **FORGIVING**/robust; “If it is too difficult, and too easy to make mistakes, they will lose interest”. (Block-based scripting like Scratch is very good for this!).
* It should be **RESPONSIVE**, ie. have instant results; “If the children don’t see the results of what they’re doing very quickly, then they will lose interest”. That is to say, we should avoid having long periods where users have to work on code with nothing else happening.
* It should have **DEPTH**; “Users will have all sorts of different skill levels – try to appeal to more than just the most experienced users”. We should thus ensure that the game avoids becoming one-dimensional and has variety in the challenges.

The teacher also took us through an introduction to Scratch, which will be detailed in section “…“ (Currently on the Market). She emphasised the range of different ability and skill levels of pupils within the school, in terms of programming. She also mentioned a large disparity between schools, in terms of the amount of focus given to programming and contemporary computer science education.

**Aims and Objectives 1.2**

/\*in the process of writing this to reflect what we have found in our research both primary and secondary

Computer science is such a broad and multifaceted subject, with many overlapping subfields and applications. In response to the question, "how can computing concepts be demystified to a new audience?"

Affecting real change means reaching homes as well as schools, and for this reason we have focused our efforts purely on software (as opposed to costly hardware) for the creation of an engaging, educational game. Influenced by David Allen’s statement (“we didn’t want people to be controlled by it, but to control it”), our aim is to briefly turn the relationship between children and digital algorithms upside-down. The ultimate intention is to facilitate the easy creation and manipulation of custom algorithms, in a visually rich, gamified context, demonstrating that computer science can be both fun and empowering.

Taking on board the (Next Gen) gaming and VISUAL EFFECTS in classrooms (our game focuses on graphics a lot)

\*/

**Planning 2.**

**2.1 Concept Development**

**2.2 Market Research**

Should section 1.3.1 not go here - shows the market research we carried out? And then what is currently on the market will go under competitor analysis?

**2.3 Initial Concepts**

# 2.3.1 Text Adventure Game

Our very first concept was based around a text adventure game whereby the player would control a character in a story and program decisions for that character, for example, programming the character to move left or right, or programming the character to defend itself against an enemy, etc. The decision the player made for the character would either progress them through the game, or (if a poor decision was made) move them further back in the game. As players progress, decisions would become harder and therefore programming would become more advanced.

The idea of this story-based game was to introduce players to programming in a fun environment of their choice; the player could choose to control a Rock Star on tour, a soldier in a zombie apocalypse or a wizard fighting enemy trolls – a variety of choices to cater for different audiences and different interests.

However, it was difficult to see where a graphical element could be added to this game. Producing enough graphics for one storyline may have been possible, but producing enough graphics for multiple stories would have been challenging and taken time away from actually developing an interesting, interactive, fun game for the audience to play.

Market research also highlighted that it was important to our target audience that a competitive element be present in the game, and other concepts that we developed catered better to this.

# SWOT Analysis

|  |  |
| --- | --- |
| Strengths  * Story element highlighted as important by target audience. * Fun, programming element to introduce audience to computer science. | Weaknesses  * Difficult to implement enough graphics to enhance game. * Lack of competitive element important to target audience. |
| Opportunities  * Can offer multiple storylines allowing catering to many different audiences ensuring game is compatible for wider audience. | Threats  * Concept has already been used for number of years; therefore game may seem outdated, especially amongst advanced games available today. * Game may not fulfill the purpose of sparking enthusiasm in Computer Science. |

# 2.3.2 CPU Mario

The Central Processing Unit game concept took a different approach as the player would act as the CPU, collecting instructions and programming the CPU to process them. The Mario concept was introduced post market research, as this ranked highly amongst our target audience.

The final concept: the CPU Mario version of the game, saw the player roaming around a grid, collecting different parts of an instruction and then, once the full instruction had been collected, decoding the instruction by placing each part in the correct order, and programming the CPU to process this instruction.

# SWOT Analysis

|  |  |
| --- | --- |
| **Strengths**  * A different style of game - the player is able to learn how the CPU works by programming each instruction as if they are the CPU. * Incorporates a Mario style game whereby the player must collect pieces of code from a grid - a concept which ranked highly in our market research. | **Weaknesses**  * A lack of forgiving, instructed as critical in our market research. If the player is unable to decode the simplest set of instructions they will not be able to process and will not be able to advance to the next level. |
| **Opportunities**  * The opportunity to appeal to different skill levels. Harder instructions which are harder to decode will appeal to students with a more advanced knowledge of programming. | Threats  * The lack of forgiving may result in a lack of interest from students and no desire on their behalf to play the game. |

# 2.3.3 Shrek Themed Game

The Shrek themed game concept resulted directly from our market research. Our target audience voted Shrek as one of their top interests, and some of the students put forth a Shrek themed mini game series where the player could play as Shrek in a number of mini games. The objective of these mini games, and how they would relate to Computer Science, was not made clear by the students that put the idea forward.

We looked into making a mini game series involving this character which ranked so highly amongst our audience; however we noted a few concerns with this concept:

1. Making one game to fit the brief from the client, that worked well, ran efficiently and interested the target audience would be challenging enough, making a number of mini games would be highly unachievable in the time given.

2. Whilst Shrek ranked highly amongst the 20 students that we interviewed, one cannot generalise about this age group based on this small percentage. Whilst their ideas and interests have helped us in learning about the age group, it seemed a bad idea to base our entire game on such a specific character. The students that we interviewed may have recently watched the Shrek movie in school, and this is why this character ranked so highly in our research, whereas students who have not been exposed to the film so recently, may have no interest in this character at all. Other cartoon characters also ranked highly amongst the students, so instead of picking one character specifically, we decided that our game should be cartoon themed to appeal to their interests.

3. Using the character Shrek could violate copyright laws as the character was invented by Pixar.

# SWOT Analysis

|  |  |
| --- | --- |
| **Strengths**     * The character featured in the game ranked highly amongst our target audience. | **Weaknesses**    * Time constrictions would make it very difficult to produce a mini game series to a high standard. |
| **Opportunities**    * Various mini games allow us to appeal to the interests and learning techniques of a range of students, if they do not learn from one game, they may relate to another much better. | Threats    * The mini game series may not be complete by the deadline given. * Copyright laws may restrict the use of a character owned by another company. |

# 2.3.4 Bee Concept

/\*needs to be fleshed out and worked in properly

we have identified an aspect of the field that is anticipated to have a continually accelerating impact on daily life, while also capable of being explained in simple terms: problem solving algorithms.

Today autonomous algorithms and artificial intelligences control many aspects of society, from weather forecasting to stock trading activity,[[15]](#footnote-21) and from transport planning & scheduling to date matching[[16]](#footnote-22) (the list is ever growing). Yet, despite the complexity of the problems such algorithms tackle, frequently the approach that is employed can be defined by simple rules that are far easier to understand. Swarm intelligence algorithms, for example, might determine optimally short paths between points by replicating slime mould or social insect behavior. Ants can tackle such problems by individually leaving a trail, signalling whether or not they have recently found food. Other ants can randomly explore, but are inclined to follow attractive trails. Over time the trails that remain stronger (reinforced by other ants) are those which are travelled most frequently. These, on average, tend to be the shortest path between desirable points, while the longer trails require more effort to maintain (tending to weaken). Hence, the problem-solving approach can be defined by just a few simple behaviors (random exploration and conditional trail setting / following). The ubiquity and real-world impact of similar algorithms, along with their definition by relatively simple rules, makes this an appealing educational topic.

Our objective is to develop a system to educate and inspire school children about the underlying mechanism of problem-solving algorithms and simple artificial intelligences. \*/

|  |  |
| --- | --- |
| Strengths | Weaknesses |
| Opportunities | Threats |

**2.4 Final Concept**

**2.5 Competitor Analysis**

**Project Management 3.**

**Overall Objectives 3.1**

The overall objectives of this project are:

1. To produce an interesting game that sparks enthusiasm in 13–14 year old, year 9 students to learn more about computer science.
2. To produce a game that meets target audience criteria.
3. To finish the game by the deadline provided by the client, producing a well polished, playable, fun, interactive end-product.
4. To produce a game that contains elements that allow the target audience to learn about the subject of computer science.

It is important to us to produce a well made, playable product; however, the product must fulfill the main purpose, which is to interest year 9 students in computer science, to get them enthusiastic about the subject, and to learn an element of computer science from the game.

**Design and Implementation 4.**

**Testing and Debugging 4.5**

We have adopted the unit testing framework check to perform testing. Check is from the webpage (http://libcheck.github.io/check/).

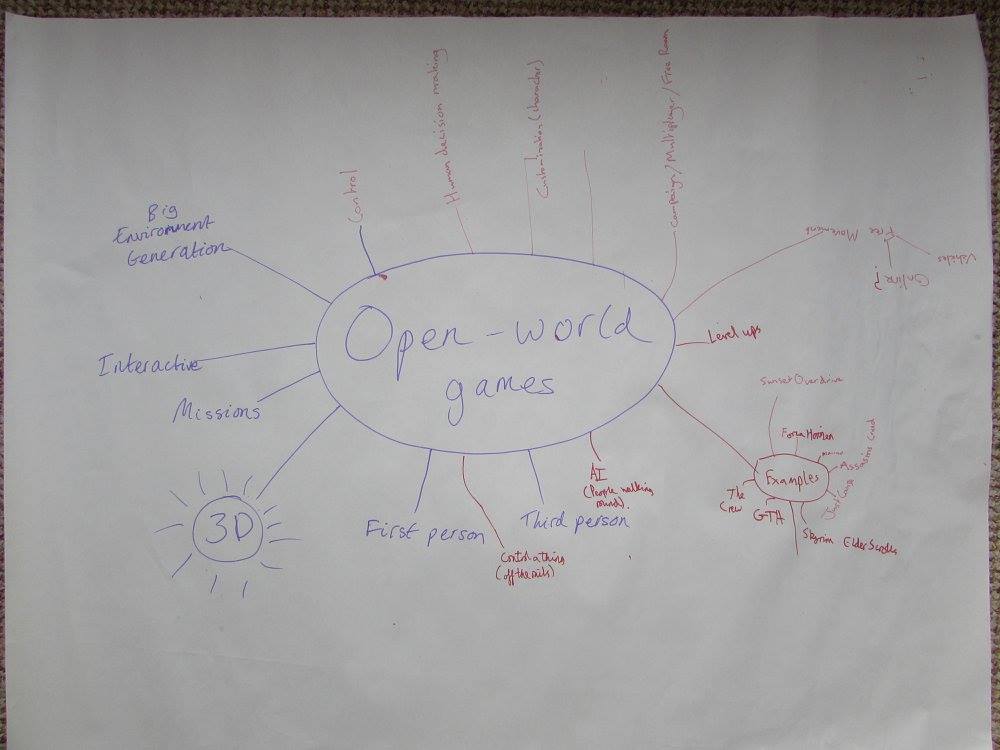
The scope of testing included (i) a full scope testing for all functions in generic.c and control.c, and (ii) limited testing on selected functions in other .c files.

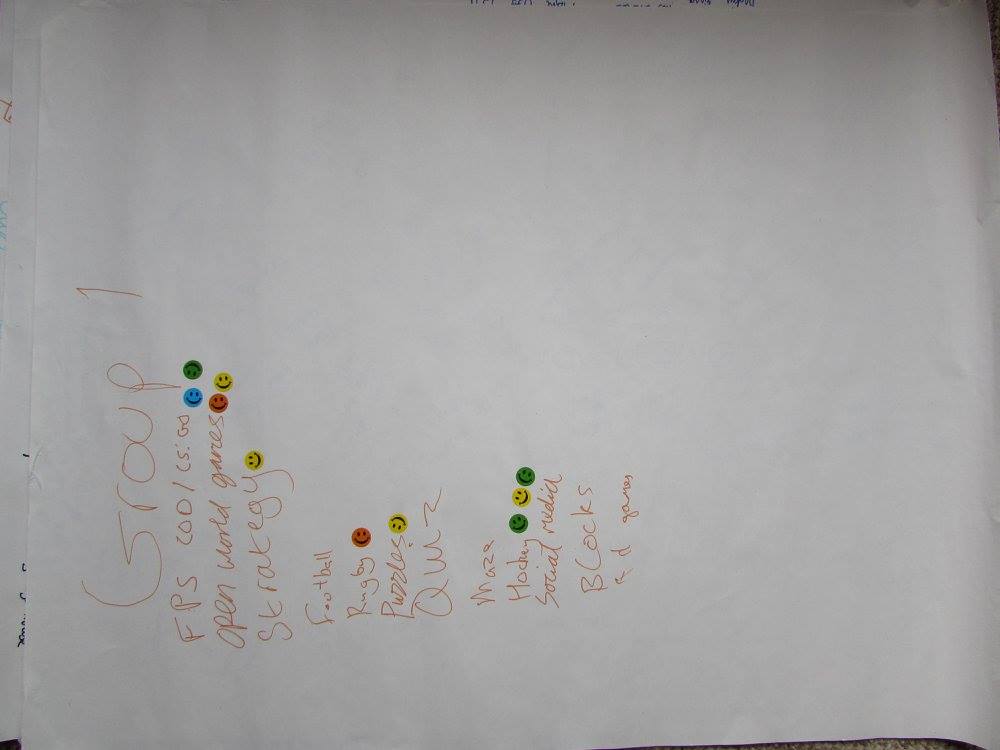
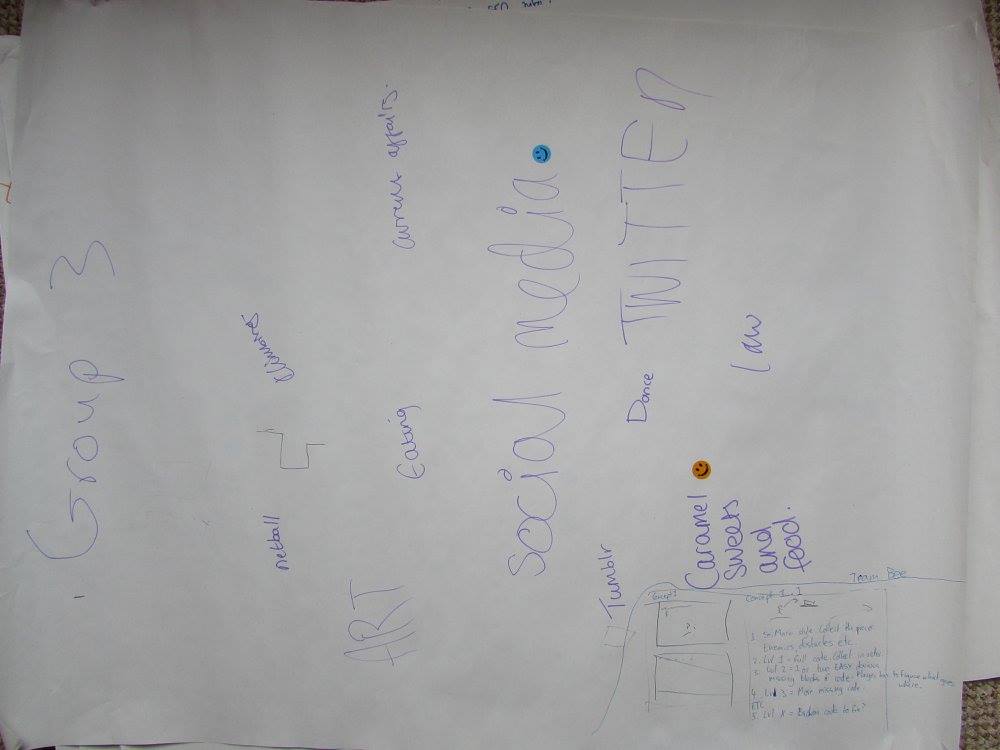
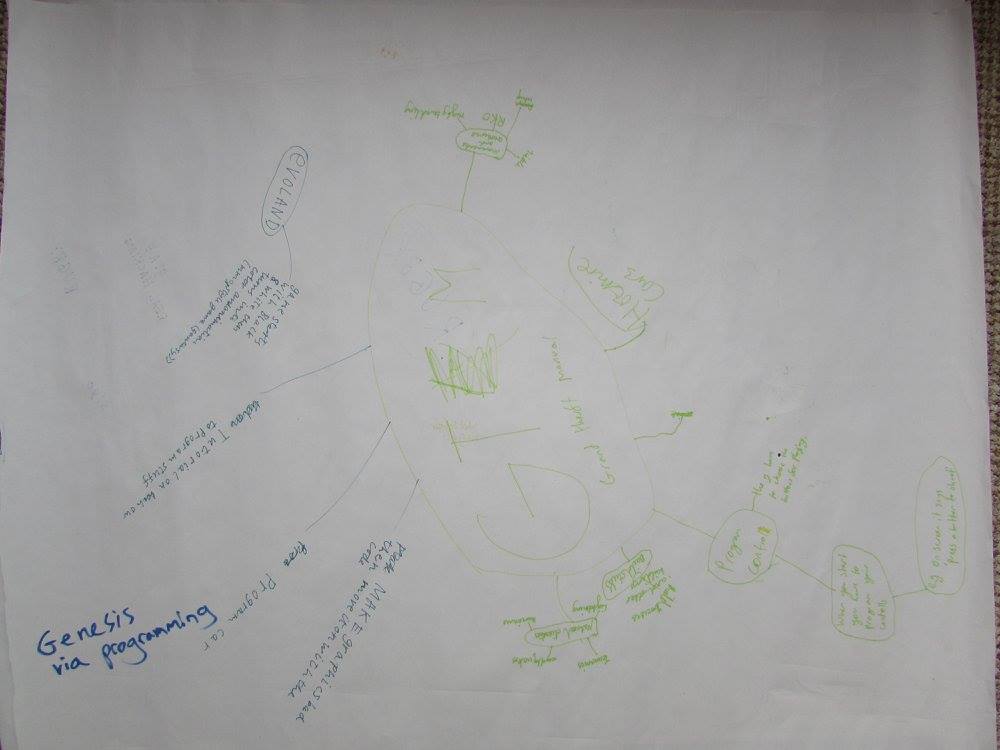
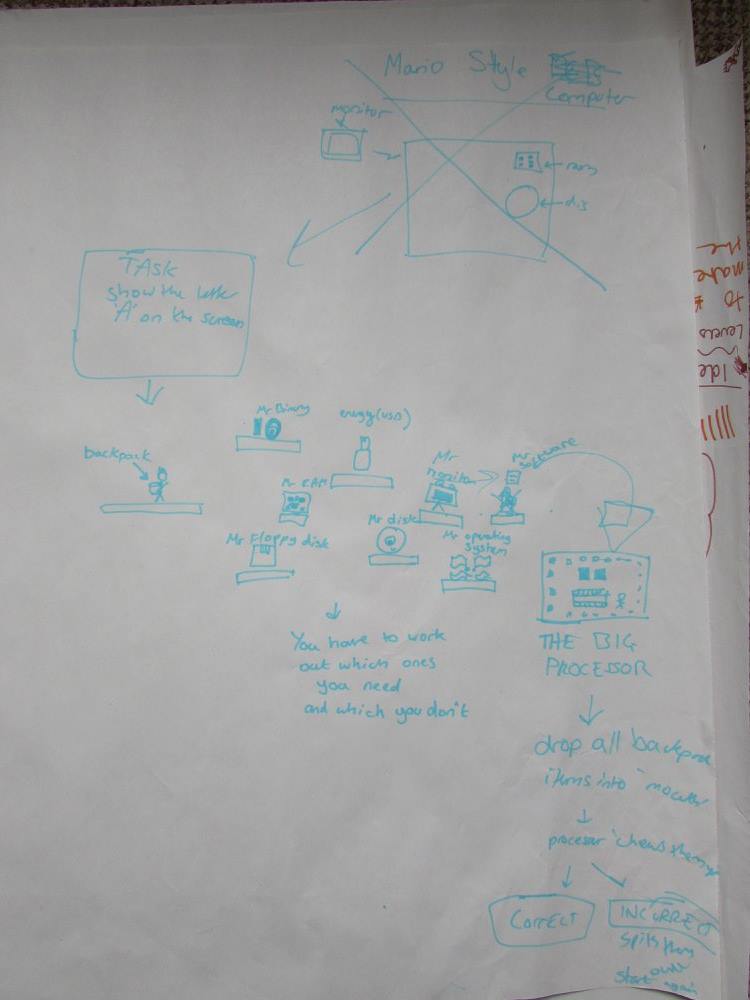
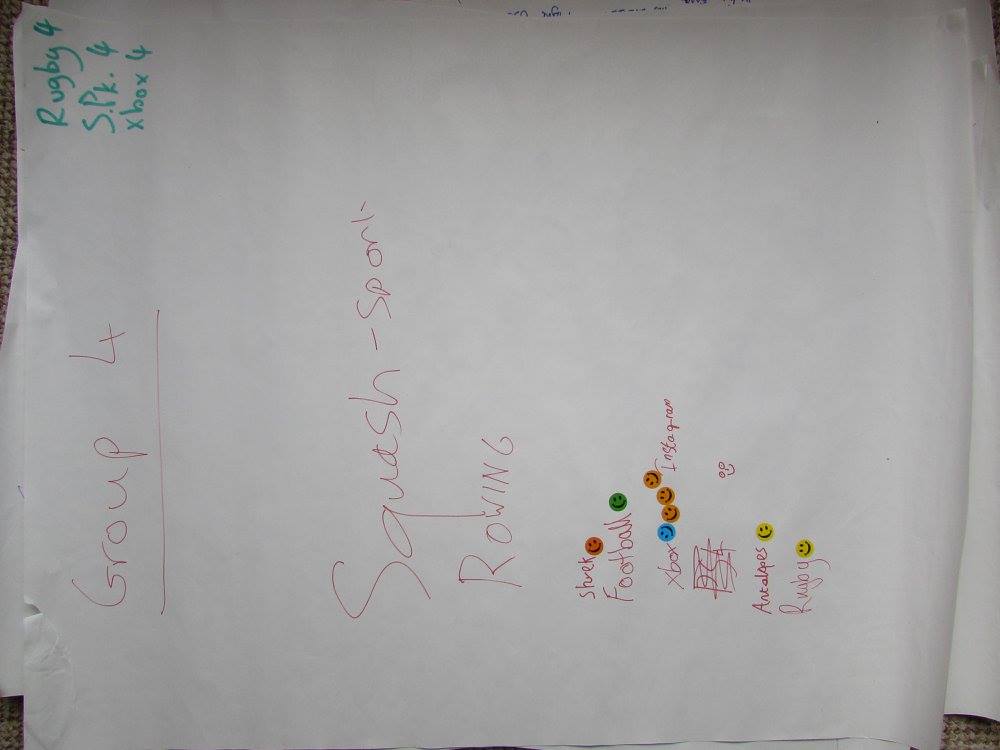
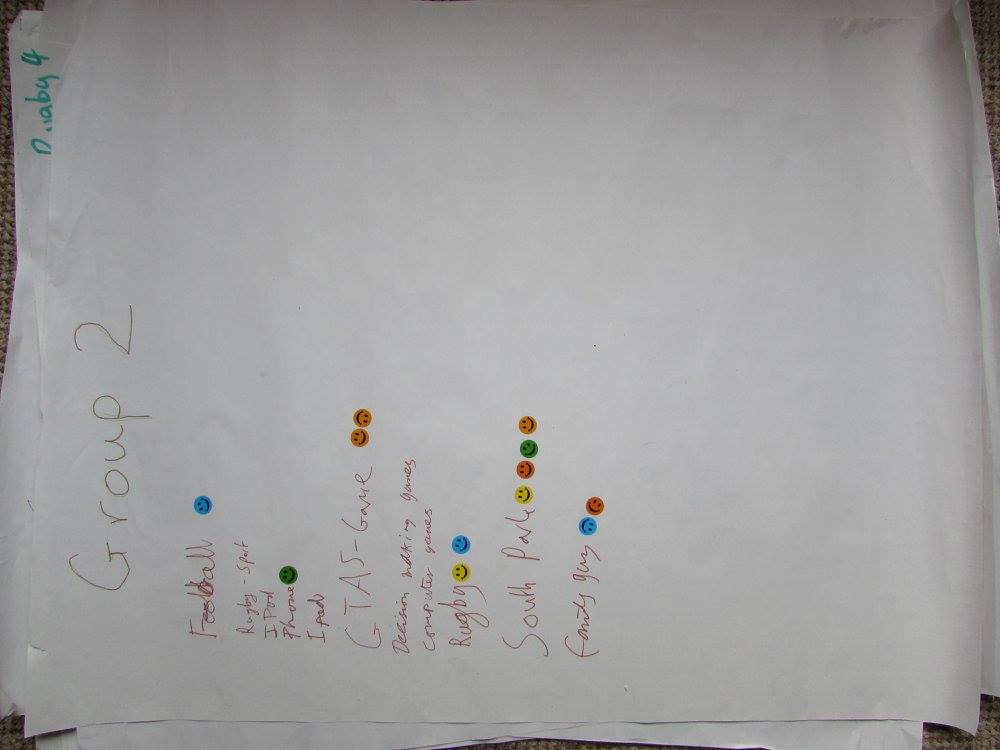
To test a function which returns an output (could be a predefined type or a struct), we would check if the output of the function match our expected value or not. If it matches our expected value, it passes the test. Otherwise, the test is failed and we need to investigate the reason of failing the test.

Some functions are void functions thus not returning any value. In order to test these functions, we would investigate the changes carried out by these functions. The testing method is to check the expected value of certain variables after performing these functions.

Some functions involved inputs and/or outputs SDL type of variables. Thus, we need to initialize these variables properly before testing these functions. For function output to the screen directly, we would need to test them using methods similar to testing void functions.

Appendix 1

****

****

1. https://www.nesta.org.uk/sites/default/files/the\_legacy\_of\_bbc\_micro.pdf [↑](#footnote-ref-2)
2. http://www.ebuyer.com/blog/2015/07/can-the-bbcs-micro-bit-inspire-a-generation/ [↑](#footnote-ref-3)
3. http://www.theguardian.com/media/interactive/2011/aug/26/eric-schmidt-mactaggart-lecture-full-text [↑](#footnote-ref-4)
4. https://www.gov.uk/government/uploads/system/uploads/attachment\_data/file/239067/SECONDARY\_national\_curriculum\_-\_Computing.pdf [↑](#footnote-ref-5)
5. https://www.gov.uk/government/uploads/system/uploads/attachment\_data/file/239067/SECONDARY\_national\_curriculum\_-\_Computing.pdf [↑](#footnote-ref-6)
6. http://www.computerweekly.com/feature/The-UKs-new-computing-curriculum-is-here-Are-teachers-ready [↑](#footnote-ref-7)
7. http://schoolsweek.co.uk/teachers-lack-confidence-in-computing-science/ [↑](#footnote-ref-8)
8. http://schoolsweek.co.uk/3m-on-and-where-are-all-the-master-teachers/ [↑](#footnote-ref-9)
9. http://www.theguardian.com/technology/2014/sep/04/coding-school-computing-children-programming [↑](#footnote-ref-10)
10. https://petition.parliament.uk/petitions/111693 [↑](#footnote-ref-11)
11. http://www.bbc.co.uk/news/technology-29010511 [↑](#footnote-ref-12)
12. <http://www.nesta.org.uk/sites/default/files/next_gen_wv.pdf> p.38 [↑](#footnote-ref-13)
13. <http://www.nesta.org.uk/sites/default/files/next_gen_wv.pdf> p.5 [↑](#footnote-ref-14)
14. <http://www.nesta.org.uk/sites/default/files/next_gen_wv.pdf> p.38 [↑](#footnote-ref-15)
15. http://www.bbc.co.uk/iplayer/episode/b03k6ypz/the-joy-of-logic [↑](#footnote-ref-21)
16. http://www.bbc.co.uk/programmes/p030s6b3 [↑](#footnote-ref-22)