Blockchain Bridge

A project by Team Inglorious Bastards for COMSM1401 Software Engineering and Group Project

**Contributions**

|  |  |  |
| --- | --- | --- |
| Name | Student no. | Contribution |
| Harvey |  | 200% |
| Leo |  | 300% |
| Nadia |  | 7000% |
| Come |  | 50% |
| Nick | nw15347 | 52% |
| Joe | Jw2 | 51% |

Contents

Abstract

Acknowledgements

**Introduction**

This section details the decisions and choices regarding development, communication and management decisions. Significant choices between tools/software used during the project will be described.

Choosing a framework

Decision on game engine made 05/02/19.

There was a lively discussion regarding which language and engine we should use for our project. A summary of main points/options discussed can be found in the table below.

|  |  |  |
| --- | --- | --- |
| C and SDL | * We all have previous experience with C. * Previous experience using SDL and C, which for nick was a negative experience. * C appears to be used for specific purposes when making games, which are not being considered in our project. | |
| C++ | * It would be useful to learn a new language. | |
| C# with Unity | * It would be useful to learn a new language. * It has 3D capability and provides a framework for * Some people felt after using Unity that there was a lot in-built, and that it would be more useful for their learning to build something from scratch. * More difficult set up methodological testing | |
| Java and JavaFX | * People felt it would be beneficial to spend more time learning Java in order to have greater experience with it upon finishing the course. * It may be more helpful for our learning to attempt to use a new language as Java is taught in another unit. | * JavaFX is used in a variety of applications outside of games, which provides useful experience if building, say an app in the future. |
| Java and libGDK | * libGDK is specifically used for building games which would make building the game easier. |
| Python and pygame | * It would be useful to learn a new language. * The example games/projects seen using pygame and python did not appear as attractive as what could be achieved with other options. * As with unity, there appears to be lots of prebuilt libraries whereas people wanted more input. | |

As the discussion developed, we narrowed the decision down to being between python and pygame, java with either javafx or libgdk, and finally unity with c#. We then outlined the most important things we wanted from framework.

* + Easy to do unit testing.
  + Less reliance on pre-built features with more built by the team.
  + Shown to be able to produce small games with medium-level graphics.

Based on these, we narrowed our choice down to using Java. The final decision to use JavaFX was based on the fact that as libGDK is specifically used for making games, team members felt that JavaFX would be more helpful to learn for development outside of games in the future.

Team Management

Agile approach:

We adopted an agile approach. Teams would work on smaller aspects of the game, however there was also a focus on integrating the work at regular intervals. This ensured that everyone was aware of what classes were available to them.

Tools for communication:

We used the project management application Slack to keep track of tasks for each team member which could also be downloaded as an app for our phones. This allowed us to keep track of who was engaged with which task, and easy visualisation of our goals, and the ability to add and update goals easily and flexibly. Our workspace was located at inglorious--bastards.slack.com.

Allocated smaller teams, often in pairs (utilising the benefits of paired programming). Pairs also regularly swapped so that each member understood different aspects of the game/code development.

Version Control

Decision to use Github / integrated with IntelliJ on 06/02/19.

Only some of the members of the team had experience with Git, however we were aware that Git usage is prevalent in industry and it would be helpful to spend several sessions in early February working on familiarising ourselves with it to use for this project.

IntelliJ has integration with Github, and we have a shared repository on:   <https://github.com/Poldigoldi/BlockchainGame>. We ensured that individual commits did not break the game, with team members creating their own branches.

Regular meetings

We held meetings each week during which we set out goals blah blah. One team member would keep minutes.

31/01/19 Initial game ideas / creating communication channels and accounts.

05/02/19 Narrowing down and developing game ideas / concept development

06/02/19 3:30 – 6pm Consolidating decision on framework

07/02/19 ?? – Further development of game ideas in preparation of presentation

14/02/19 1pm – Presentation of different game ideas to cohort and final decision

19/02/19 ?? – Delegation of pairs for paired programming to different aspects of game

28/02/19 – Progress report from the different pairs – continue working on delegations from last meeting.

Concept development

Seeds of Thoth

Concept art:



**Target audience: 12+**

Educational aspect:

* Gain an in depth understanding of different computer architecture components.

Description:

* Two-dimensional game with a scrolling background, in a similar style to kingdom two crowns (top left).
* Electronics are collected through exploration, taken back to base and combined to repair a crashed spaceship.
* An underground base with room for expansion, to facilitate user creativity in a similar vain to oxygen not included or Terraria (see other two images)

Possible Additions:

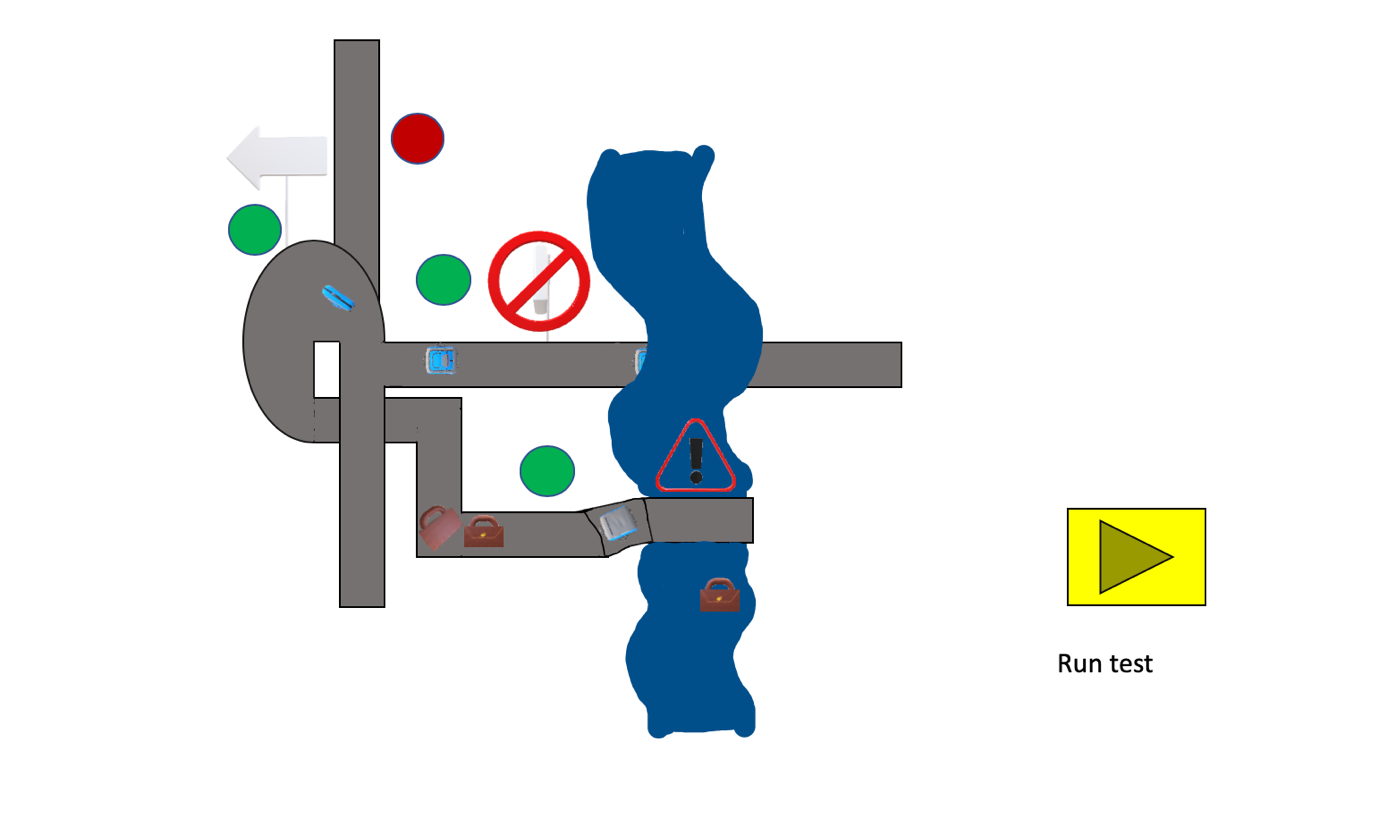
* Non-playable characters, for example settlements allowing for more interesting acquisition of electrical components.
* Base defence and management
* Increased complexity of the computer architecture
* Multiple worlds which to explore – assuming the ship has a tendency to crash.

Decision to drop:

Positives for the game include the fact that games in this style are typically very immersive and are aesthetically pleasing. However we felt it may have been ambitious to create a large, vivid and interactive world, and it is difficult to reconcile with the educational aspect of the project.

Deliver The Package / Cliff Clamber

Concept art:



**Target audience: 7+**

Educational aspect:

* Introduce users to the concept of version control, as well as if statements and for loops.

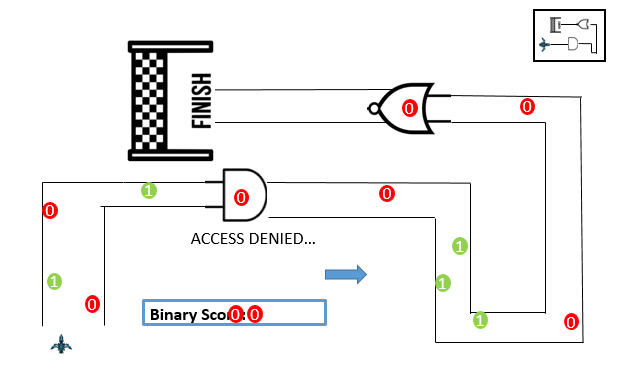
Description:

* Top down management game. 2 players are assigned with the task of transporting objects on a conveyor belt to designated destinations.
* Different hazards obstruct successful delivery.
* One player is responsible for inactivating specific routes as to not lose packages. Whilst the other is able to edit a copy of the map adding little bits of code to automate the process for example IF blue suitcase turn right. Amendments are can be tested to check if they work in the intended manner.
* Any changes must be approved, by the other player before the game board is updated.
* To make updates money is required, money is made by successfully delivering items.

Possible Additions:

* Increased complexity of modifications that can be made by player 2.
* Procedurally generated hazards, so that no map is the same.
* Rather than conveyor belts the setting could be of herding animals over a perilous path.

Hard-drive

Concept art:

**Target Audience: 11+**

Educational aspect:

* Learning about different levels of computer architecture, including logic gates, transistors and building upwards towards basic computers.

Description:

* Players race each other along tracks designed to form logic circuits.
* Speed boosts can be achieved by correctly following the logic pattern associated with each gate given a particular input.

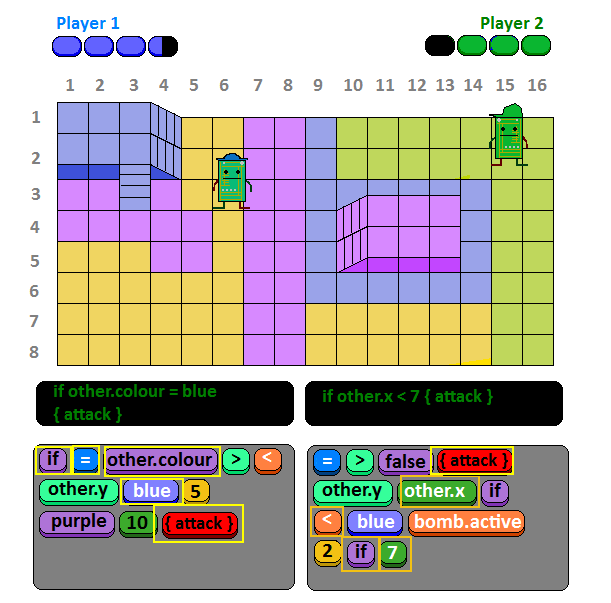
Possible Additions:

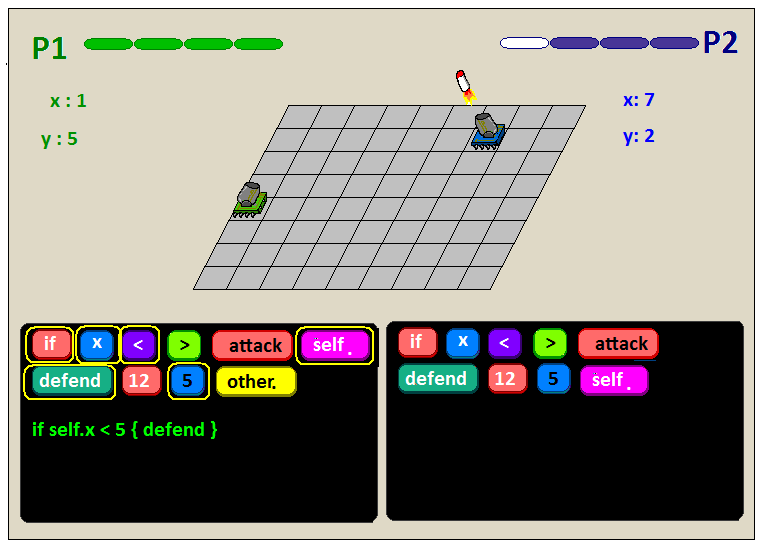
* Items that can affect the player / other players.
* Levels/tracks with different layers of abstraction, e.g. a memory chip (SRAM/DRAM) with cells.

Decision to drop:

Positives for the game include the fact that we felt it was easy to relate to many different aspects of computer architecture that we previously covered. However both previous years example’s showed that other groups had very similar ideas, with a similar graphical style to what we wanted to do. We also felt that although it incorporated computer science ideas, it was lacking in its educational aspect, and we could be more creative with what we would teach our audience.

Battle-chips – Presented on 14/02/2019

Concept art:



**Target audience: 7+**

Educational aspect:

* Become familiar with different kinds of statements (e.g. if, while), variables, and grids.

Description:

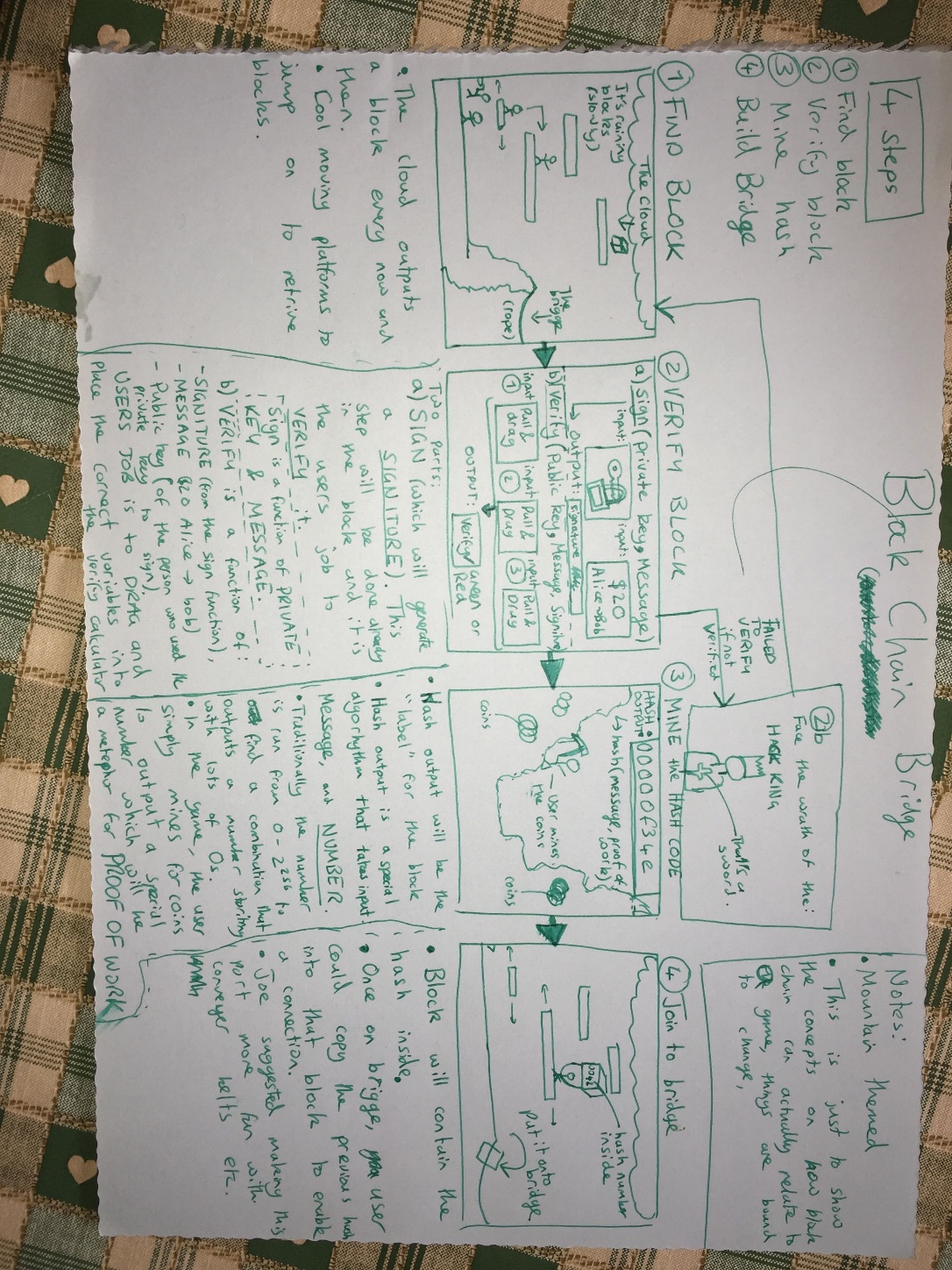
* Terminal displays the code that the player attempts to type out, upon success they can attack the other player.
* Chips can attempt to move out of the way but this is slow, your best bet is to keep the pressure on your enemy by bombarding them with attack statements.

Possible Additions:

* Items, more variables.
* Interactions with environment
* The possibility of coop versus AI/computers or solving puzzles together using similar logic.
* More maps / maps with a comp-sci theme.

Feedback:

Around 10 people from the cohort voted for the game. Ian gave some positive feedback regarding the graphical choices and simplicity of the game.

Block chain bridge – Presented on 14/02/2019

Concept art:

**Target audience: 11+**

Educational aspect:

* Teaching the audience an overview of how blockchain works, using three different methods to teach the stages of verification, hash codes and distribution.

Description:

* Mario style / platformer style world where the user collects blocks. Upon collecting blocks they are faced with challenges that teach them about blockchain. On success they add this to a bridge to signify progress. Challenges involve mining for hash codes for proof of work and so on.

Possible additions:

* Adding enemies to the platformer such as hackers
* Adding tooltips which teach about the game, and the user may be tested on.

Feedback:

Around 15> people from the cohort voted for the game. Ian gave some positive feedback regarding the educational content of the game and potential. In particular the subject matter appeared to be relatively novel for game development in the course.

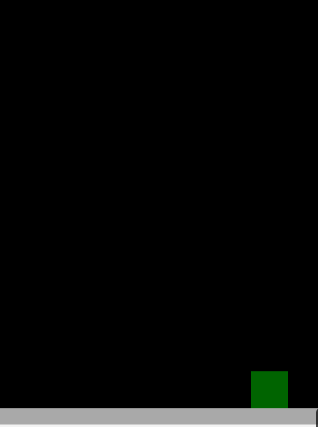
**Game development history:**

1. **Meeting: 19/02/19**

Set out initial goalposts for development and allocated team members to work on them. Three paired programming teams were decided to work on each initial goalpost.

* Basic platformer map (walls, floor, items, ladder, enemies) – Harvey / Joe
* Main player (movement) – Come and Leo
* Minigame screen (with ability to drag elements) – Nadia and Nick
* Basic game menu

Nick made the walking animation for the main character.

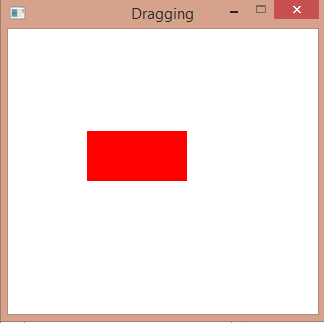


1. **Meeting 28/02/19 10am**

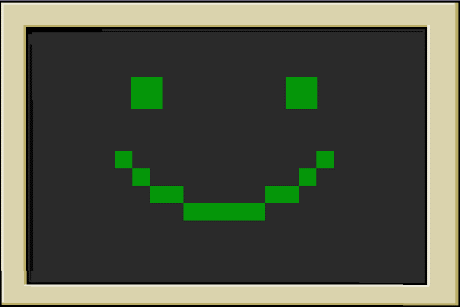
* Come and Leo have followed tutorials to create the basis for a platformer with graphics using Javafx. They also created an item object that can be picked up and moved by the main player, which is capable of moving using keyboard input.
* Harvey and Joe discussed how to make a level and have worked towards creating the map that will incorporate everyone else’s elements.

Design considerations:

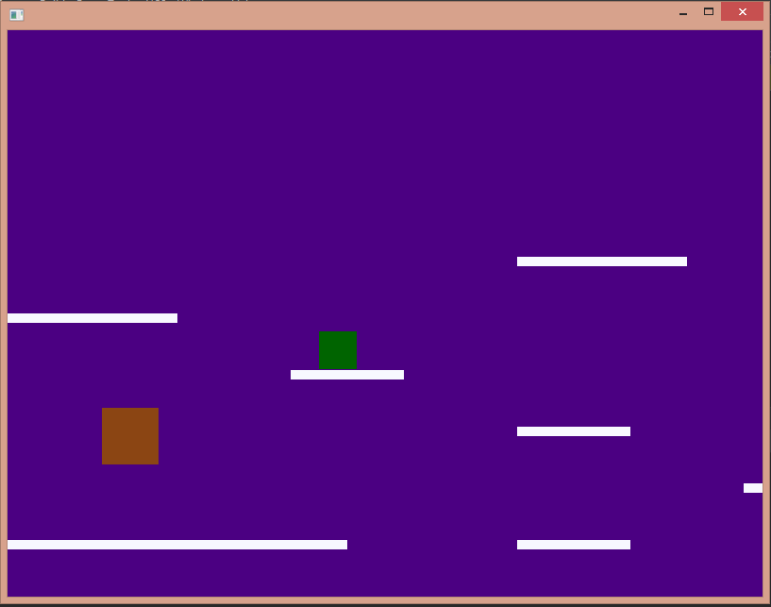
Create game map using 2D array in order to make it easier to plan the levels.



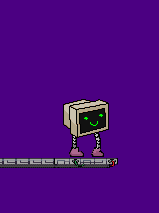
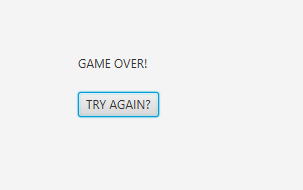
* Nick and Nadia have created a draggable object which can be dragged and snapped onto a target object which will form the basis for the minigame.

1. **Meeting 26/03/19**

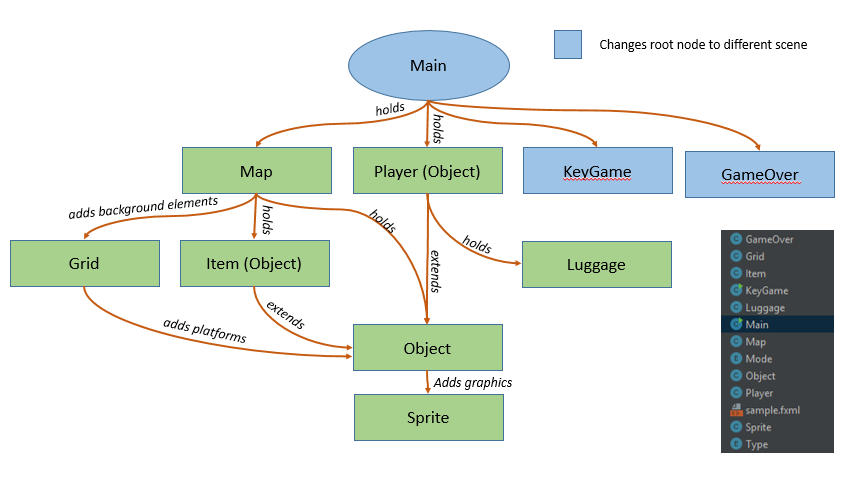
* The teams all met up to integrate what we had done so far. At this point when a player picked up a block, there was a successful transition to the minigame ‘screen’, and a successful transition back to the game when the minigame is ‘completed’.
* Nick worked on the transition screen (aesthetic) to the minigame by working out how to animate in Javafx (see X).



1. **Further developments during April**

* During the Easter break further developments were made and are being tracked via the group communications.
* Come and Leo, Joe and Harvey integrated the Level ‘Map’ for the player to move around (see X). Game over screen, after detecting death (falling out of the level) has also been implemented.
* Nicky added the Sprite Class, which attaches an imageView to an object that updates, allowing walking animations for the character, although this cannot be demonstrated in this document (see X). He’s also added backgrounds and different types of platform blocks.
* Nicky abstracted several classes by producing the Object class, objects are a ‘Node’ group, containing a collision box and an associated graphics. Players, items and platforms are all objects with their own methods attached.
* Nicky added clouds and cube graphic.
* Nadia developed a minigame class which contains drag-able objects that detect when they are overlapping.
* Nicky added layers to the graphics to allow for some parallax scrolling.



Outline of currently used classes as of 13/04/2019 for reference****