Forklift Frenzy Group Report

Alexander Haining 40283659
David MacGruer 40172073
Matias Nilla Matias Rafael Malmivaara 40286188
Matthew Newbigging 09004805
Marie Pearson 40181382
Zoe Wall 40182161

May 3, 2017

Contents

1 Abstract

This report details the progress made in completing the Forklift Frenzy group project, illustrates the various learning outcomes achieved and compares original aims with resulting deliverables.

Forklift Frenzy is a 3D video game for use on Windows PC, suitable for an audience of all ages, created from original assets only. The game lets players take command of a number of various forklifts, with which they must navigate a map to retrieve and deliver certain packages for points, against the clock.

The Group Project module provides the perfect opportunity to allow a group of students keen to work on games in their future careers to create a portfolio piece, whilst gaining some insight and experience in working in a small team to publish a game created from scratch. As such, Forklift Frenzy is a student led project, with the project manager acting as client. The roles and outcomes carried out by group members include:

- Project Manager, client and programmer - successful management of the software project; documentation, risk mitigation, feature definition, minor development tasks
- Lead Developer - all major development tasks, version control
- Artists - asset creation to specifications; 2D concept and UI art, 3D models, textures, adhering to agreed asset standardisations
- Sound Designer - creating and implementation of all sound effects

Described in detail further on in this document, the success of the project relied on overcoming several key obstacles:

- Agreeing on a work pipeline suitable to create usable game ready assets
- Appropriate collaboration in having multiple artists and programmers work on similar tasks and the resolution of any conflicts
- Thorough understanding of technologies used

The main deliverable for the project is the executable for the game itself, showcasing the work done by all team members as outlined above.

2 Background

Having an interest in working in the games industry, yet no real experience in doing so, this module seemed the perfect opportunity to do just that. Before the module began, last term, emails were sent to certain lecturers looking for recommendations of students who, while adept in their field, also possessed a similar interest to work in the games industry. Following brief interviews, the team was formed in which each member wished to use the module as an opportunity to create a portfolio piece. As such, the main aim of the project is to make available a playable game which demonstrates each team members proficiency in their field; game models and textures from the artists, original sound effects from the sound designer, game logic from the programmers and the successful handling of a software focused project by the project manager.

Second to illustrating team members individual skills, another important learning outcome of this project is to demonstrate the ability to work together effectively, with those of different disciplines within the team, to have the project meet with success. Meeting this aim should result in a better understanding of an individuals place in a group, and in putting systems in place to allow better collaboration and work flow between everyone involved.

2.1 Context

Forklift Frenzy is best put in context by considering its main purpose; a portfolio piece, rather than a means of making money. This software, while it will be used by the public, is not purposed to solve a particular problem, or have any use other than entertainment. It is not absolutely essential that the software runs bug-free, though it is of course preferred.

2.2 Scope

The scope of the project is such that only one operating system, Windows PC, will be targeted due to a focus on content and not compatibility across multiple platforms. The game itself will only include one playable level, the benefit of which is to require a minimal number of assets before the project may be called a success. This is largely due to the fact that game-ready assets take time to create, thereby limiting the number of assets that may be included given the short project time frame. The deliverable is intended to be a demo, or proof of concept, rather than a fully fledged game.

2.3 Limitations

Time available is a considerable limitation to the project, the lack of which resulted in the original project goals being revised midway through the module. Lasting only one semester, and running alongside two other equally demanding modules, each members available time is never guaranteed to be the same every week throughout the project. This affects the number of tasks that may be completed throughout the project, meaning that the list of features that must be implemented before project end has to be concise, and monitored throughout to ensure that there is indeed adequate time to complete them.

A close second in terms of limitations is software experience, which impacts on time; many team members were interacting with the software used for the first time, resulting in a longer development cycle than might be thought of as typical for this sort of software project. Whilst each team member is proficient in their own field, there was at project start less overall experience in collaboration; storing and sharing work, ensuring assets passed on are usable by programmers, maintaining appropriate communications. This results in more time being spent working out effective systems and standardisations, further limiting the amount of tasks completed.

2.4 Aims

The original aims of the project were to have the game feature:

- Single Player; a game mode which can be played without an internet connection. Allows players to complete missions to earn in-game currency, which can be redeemed to buy other types of forklift.
- Multiplayer; a game mode requiring an internet connection, in which up to six players on two teams compete on the same map to collect as many boxes as they can, with the winners having the most boxes collected at the end of the round.
- Menu System; consisting of a main menu, pause menu and forklift selection screen, allowing users to navigate to different game modes, edit options and quit the game.
- Game state persistence; via loading and saving local player data. This allows players to continue from where they left of in previous playthroughs, maintaining their high scores, currency and forklift unlocks.
- Four forklift classes; with distinct models and characteristics for each.

Further to the above points, at project start it was intended that the game be submitted to Steam Greenlight. This process involves publishing screenshots, videos and a description of the game to Steam for users to view. Users then have the chance to upvote, or downvote, the game depending on whether they would like to see the game in the store to play for themselves. Should the game be the most voted at any time, Steam allow it to be published, and sold, through their store.

3 Results

As mentioned previously, a chief goal of the project was to adopt an effective work flow between team members given the difference in disciplines present in the team; creatives, organiser and logical thinkers. The artists and sound designers were in charge of creating the assets that would appear in the game, while the programmers were responsible for building the logic and compiling the completed assets within the game scenes. This meant that tasks completed by the artists and sound designer flowed through the programmers, who also acted as testers.

Underpinned by communication, the overall pipeline put in place allowed work to be shared across the team with relative ease; tasks were worked on individually, with team members being updated on progress or problems throughout the week. There were slight differences in the work flow for each discipline, as outlined below:

- Art: After being set the task of creating a certain model for the game, artists could work individually. On completion, the model was passed on to the lead programmer for integration into Unity, where it was also tested to ensure acceptance criteria was met, which includes meeting the agreed upon standards. Should any criteria not be met, the model would be sent back to be updated.

- Programming: The numerous scripts necessary to create the game logic were developed in a certain order based on the scenes as they were made in Unity. Programmers would work on separate scenes where possible to make merging work simpler. Completed tasks would be first tested in Unity before a pull request was made to update the master branch with the latest work.

- Sound: The sound designer was able to work individually for the most part; recording and editing sound clips formed a large part of the sound development cycle. Once the sound banks were made, the implementation required a programmer to edit existing code in order to add sound into the game.

3.1 Summary of Changes

Given that the development time of certain assets, and completion time of some tasks, were uncertain at time of planning, extra time was allotted. Despite this, midway through the project it became clear that the original goals were too ambitious to complete before project end; there simply wasn't enough time to complete everything to standard. In order to ensure better quality and completeness in the final deliverable, a couple of goals were dropped from the project aims. (reflected in MSCW - put here or in PM?)

The first goal dropped was the multiplayer mode; this is a large undertaking in itself, and can be considered a separate game from single player due to numerous differences in game logic. It requires learning a different coding approach, networking, of which the time taken to achieve is uncertain.. At the time, there existed many bug related tasks to do with single player that were previously unaccounted for, so the decision was passed that multiplayer be dropped in favour of focusing on polishing the single player experience. (worth mentioning here that we said we'd work on it after project end, on our own time?)

Another goal set aside was the submission of the game to Steam Greenlight. Unfortunately, late in the project Steam announced that they would be replacing Greenlight with another, more costly alternative process. As such, this aim was dropped since the group did not wish to incur the cost to publish what is effectively a proof of concept rather than a full game.

3.2 Outcomes

With regard to each of the above outlined aims, this section compares the results achieved as seen in the final game.

Single Player Mode

The vast majority of the work done throughout the project was put into the single player mode; most assets, sound clips and scripts created are showcased in this scene.

Figure 1: Single Player Screenshots

This image was taken during single player gameplay midway through a mission, illustrating the forklift having picked up a box and heading back towards the drop off point. The map itself is a warehouse, primarily filled with shelves and boxes, alongside other miscellaneous objects.

Figure 2: Clock In Approach

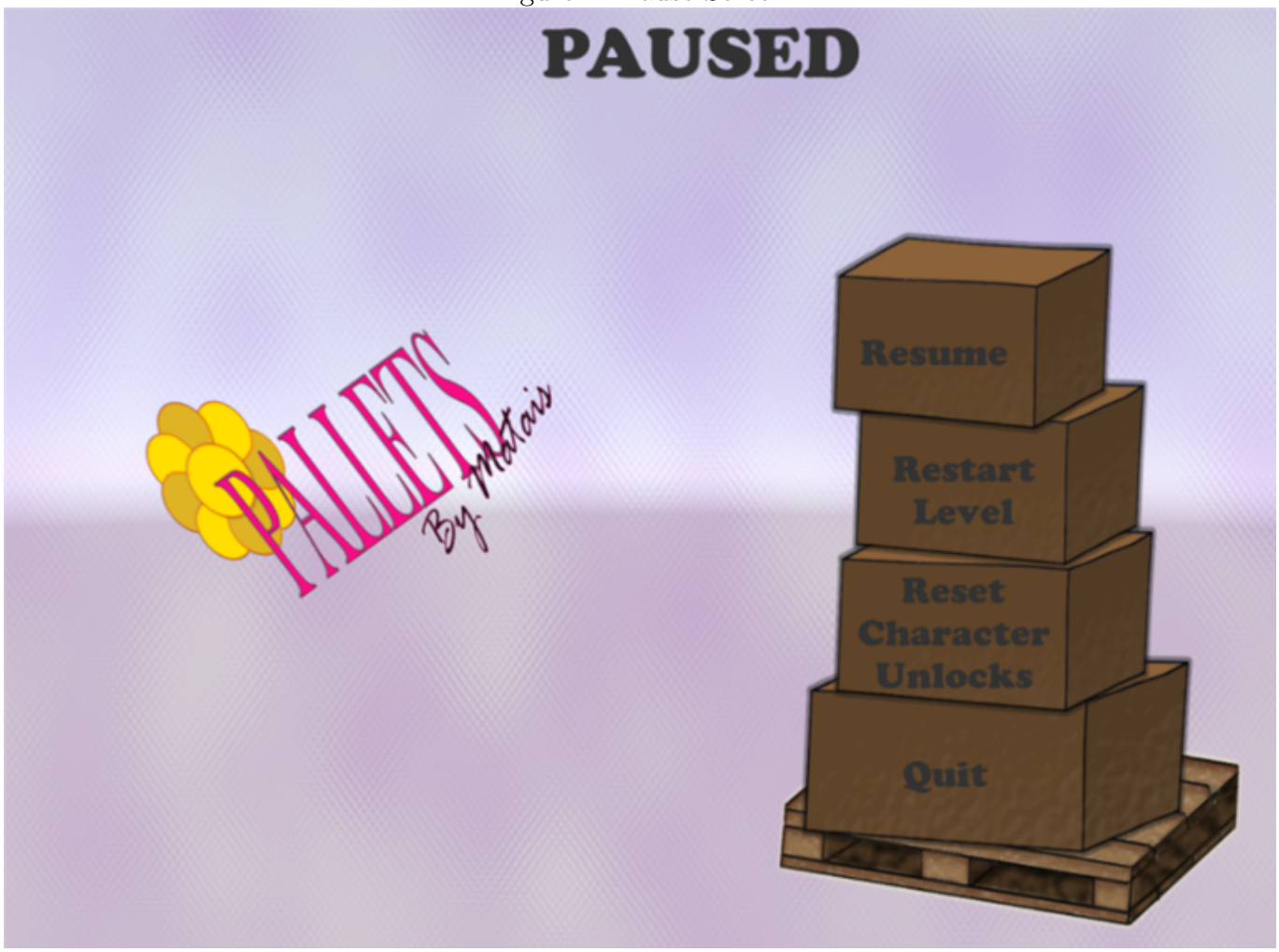
The Clock-In object is how players begin missions within the game; by driving over the red circular activation pad on the ground in the spawn room, players can decide the difficulty level of the mission which affects the number of boxes spawned.

Figure 3: Clock In Close-up

A better view of the clock-in model; this clearly shows the in-game user interface with which the players interact to start missions.

Menu System

Figure 4: Pause Screen

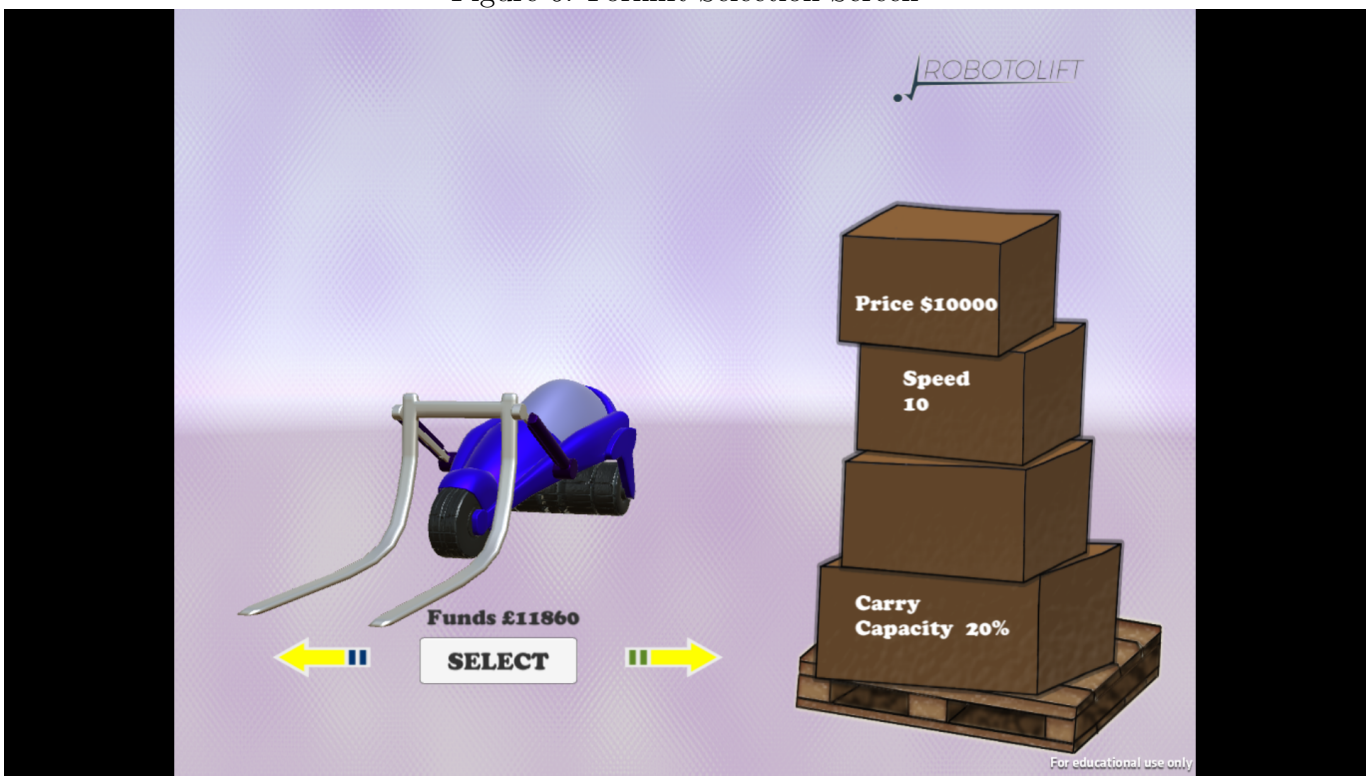


The above image displays the in game pause screen, which has several options the player can interact with. The image shown left of the boxes is chosen randomly from a set, each time the pause screen is opened.

Figure 5: Main Menu

The game begins by playing the main menu intro animation; the above forklift drives from screen right into centre screen, with the menu items appearing over the boxes once it stops. Throughout the animation, and whilst in main menu, the game music is playing.

Figure 6: Forklift Selection Screen



The forklift selection screen can be opened whilst the player is in the starting room, containing the clock-in object. It shows all four forklifts, their statistics, the players total funds, and allows the player to buy or select the currently displayed forklift.

Persistence

Figure 7: Persistence Screenshot

The above image demonstrates the persistence aim being met; the high score value bottom middle is saved despite closing/reopening the application. As can also be seen in the above forklift selection screen image, the player funds and unlocked forklifts are also maintained over multiple playthroughs after having closed the game.

Four Forklift Classes

Figure 8: Forklift Classes: Redneck (L) & EZ-Reach (R)

Figure 8 illustrates two of the four forklift classes; Redneck and EZ-Reach models.

Figure 9: Forklift Classes: DinkyDink (L) & Robotolift (R)

The above shows the other two forklift classes made for the game; DinkyDink and Robotolift.

3.3 Pipeline

After collaboratively working out a plan for features the implementation phase of the project could start. The working pipeline consisted of a similar structure to games development within the industry from concept design, pre-production and onto production. Unfortunately due to the limitations faced during the project, the post-production stage of the development was not fully completed.

Concept Design The original concept of the forklift game was conceived before the start of the Group Project in January. This gave the team plenty of time to iron out the designs.

Pre-Production During the initial stages of the project, the team had many group meetings for collaborative idea generation. This proved to be a very useful and productive stage of the process. Documentation was created that can be seen in figures 13-17 within Appendix D, this detailed such things as level design, user interface design, mechanics and character design. Code design documentation was also created detailing the classes needed and the required relationships between them.

Production Once the design documentation was completed, the production stage could begin. Due to the nature of the tasks, the group could be split into three separate sub-teams to develop artwork, implementation, and sound separately. This model for working although suiting the project proved to be difficult as times as each members work flow relied on each other during the final stages of the project. This seemed to lead into more complex technical issues that were not foreseen in the projects output.

Artwork

The task for the 3D artists was to produce game ready assets for the developers to implement into the Unity game engine. The process of producing the models would start with the concept artist delivering the required reference images. Next would be to develop the model from the concept images using Autodesk Maya. Once this process was complete the models will be re-topologized to ensure the polycount is correct and the level of detail remains sharp. For proper integration into the Unity game engine, the models needed to be structured with a naming hierarchy that worked for the game developers. During this stage, the asset needed to have the local space coordinates formatted correctly to be controlled using code within the game engine.

The model was then sent over to the lead developer for preliminary testing and would be sent back to the modellers for any final edits and adjustments. Once completed the asset was now ready for creating UV (texture) maps. This process is to unfold the external material correctly over the 3d asset before taking the asset into Substance Painter where they could be textured to the desired style. Once the team had finalized the texturing, the models were ready to be exported and packaged as an FBX file with the texture maps. From here the Game developers would be able to properly integrate them into the game engine.

$$Concept\ Art \rightarrow Model \rightarrow Re-Topology \rightarrow Naming\ Hierarchy \rightarrow Formatting\ Coordinates \rightarrow Testing \rightarrow Adjustments\ Edits \rightarrow UVing \rightarrow Texturing \rightarrow Exporting \rightarrow Game\ Ready$$

Code

The task for the development was initially split into two parts; the lead developer worked on the initial game logic and code base for the mechanics of the movement within the game following the proposed class structure, see Figure 20 in Appendix D; the other developer worked on initially creating the menu elements to avoid working on the same scene at once which causes problems with merge conflicts.

Learning how to use version control through git was one of the main learning objects of this project, which required the whole team to learn how to use git effectively on a collaborative project. A repository was hosted on GitHub, and the lead developer attempted to teach the team how to follow a standardised feature branch workflow. Each person developing on the project was to create their own working branch and merge to master when appropriate keeping the master branch release ready, with no compile errors. This began to get difficult for some of the members of the team so it was decided that when a merge was needed, the team member would issue a pull request that would be overseen and any merge conflicts or problems could be fixed by the lead developer.

As for the logic structure of the game itself, most of the development teams work had to be iteratively improved as and when the new models were finished by the artists. Their work included: scaling and transforming models correctly to tie into the logic of the code, work with the unity physics engine to attach colliders and rigid bodies to allow the models to interact appropriately, design and implement simple user interface features to allow for the user to control the game, and ensure that the single player prototype was a playable game. Initially research was done into the multiplayer aspect of the proposed game, however due to project scope and time constraints this did not end up in the finished product.

Sound

The sound designer/engineer used the middleware FMOD Studio to make assets that were created game ready to plug-in straight to Unity. Pro Tools was used as the main digital audio workstation, most of the work in achieving the sounds the group needed for whatever purpose were recorded and edited at this stage. This process took several recording sessions, recording all sorts of different materials from cardboard boxes and sticks to heavy metal items in 24bit and 44.1kHz, to have full dynamic range without digital errors that sometimes occur in digital to analogue conversion. These clips were then edited be conservative for the sake of memory usage and such requirements for the audio to be game ready.

To create the final sounds heard within the game, each clip was layered within an FMOD editing session, for example, the sound of the forklift engines were clips of toothbrush and vacuum cleaner sounds that were processed heavily, using such effects as equalisation, pitch shift, and distortion to name a few. The sound clips also had to be looped seamlessly as the game is a real-time simulation the sounds must be continuous and cannot just stop abruptly.

The most important part of this stage of development was communication between the sound designer and the developers to implement the audio within the game. Documentation that can be seen in figure 9 from Appendix D details some of the more technical specifications for implementing the final sound clips.

3.4 Testing

Types of testing undertaken throughout the project include mostly acceptance criteria testing, with some explorative testing. Testing sessions were planned from project start; aimed to coincide with the completion of major development tasks. However, due to the uncertain nature of the duration of some tasks, testing took on a more ad-hoc nature as the project matured.

Acceptance criteria testing involves comparing the requirements of a development task with the task deliverable, visually checking that the implementation was correct. Some assets required more testing than others; sound tasks required little testing since after the programming implemented the sounds in game it was a case of hearing if the sound effect played or not, after which it was tweaked to the desired effect. Assets created by artists had to meet the agreed upon standards; naming conventions met, appropriate item groupings and parent/child transform relationships met. This was done by loading the asset into Unity, checking its meta data and viewing the model within the game. Should any criteria not be met, the item was sent back to the owner for updating.

Explorative testing involves thorough use of the product in as many situations as possible. In this case, that means playing the game through is as many ways as can be thought of. This reveals issues and bugs perhaps previously undiscovered that may have been the result of a knock-on effect after a merge of work, or some change to the game logic, or perhaps an effect that was unaccounted for. While this type of testing was never planned, it was undertaken following acceptance criteria testing for a brief time, and at semi-regular periods throughout the project for a longer duration. Any bugs or issues found through explorative testing were added as issues on Github, where they were later prioritised and included as tasks where appropriate. These issues tended to be minor, requiring only a small amount of time to fix; the final week of development was dedicated to solving these issues.

4 Project Management

4.1 Approach

Throughout the project, certain characteristics of the agile methodology were put to use, including:

- Close communication and collaboration between all team members throughout, to ensure any problems that arise are dealt with swiftly and with greater efficiency.
- Iterative development by means of working in weekly sprints, with clear goals set per sprint, and stand-ups taking place at the beginning of each weekly meeting.
- Ensuring flexibility by adjusting task requirements or project goals on the fly.

Adopting the above helped to maintain motivation amongst the group; communication aided in understanding each member's skills and progress, so no one member was left unaware of what others were doing. Working iteratively gave way to quick development cycles since the work done needn't represent a perfect, final product; previous work was improved upon with further tasks, the result of which saw improvements being made more often. Throughout, some tasks' acceptance criteria were adjusted where necessary, and project goals dropped midway, which helped the group to gain focus on the more important, must have, goals.

The MoSCoW method groups all project features into four categories; must have, should have, could have, and won't have. The group's MSCW document proved useful to determine the priority of all features, clearly illustrating the most, and least, important aspects of the product. This was updated after any project goal was re-prioritised, or dropped - such as midway through the project when it was decided that the multiplayer mode and submission to Steam Greenlight were no longer must have goals. This afforded the group a clear view of the latest opinion on each feature.

The project manager/programmer acted as client for this project, which allowed direct communication, quicker decision making, and a technical understanding of the development process. This resulted in saving a lot of time that would otherwise have been spent meeting with an outside client or waiting for decisions to be made, thereby catalysing the development process.

Further to the above, project risks were recorded in the risk register at the start of the project, and kept an eye on throughout. Issues faced later in the project were of a more interpersonal nature, and so dealt with privately on an ad-hoc basis. Backup plans were put in place to cover any work unable to be completed as a result of these issues.

4.2 Collaboration

Various platforms were put in place to allow team members to communicate and collaborate on work with one another. At project start, several pieces of software were available for team use:

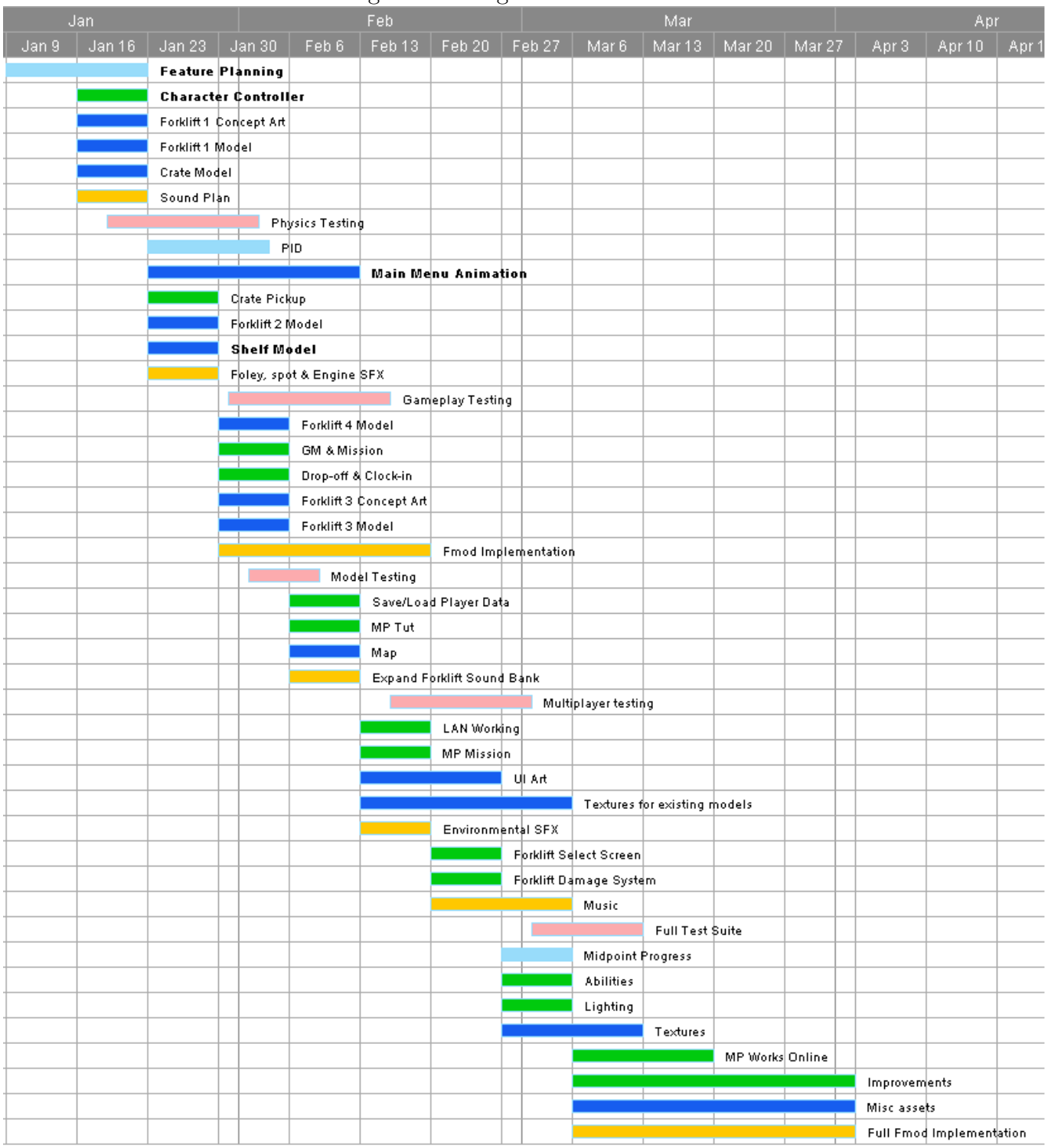
- Slack: Work-based messaging software, available on mobile and PC. The team had its own group that allowed users to discuss various topics (art, code, sound, ideas etc) on individual channels. Files were also shared through Slack.
- Google Drive: The primary source of storing all files to do with the project. All members had access, allowing them to make backups of their work. Contains everything from photos of wireframe doodles, project management documents, to assets. Proved invaluable in having one place to store all work.
- Github: Used primarily for the scripts created by programmers and sound designer for integrating FMOD into Unity. Allowed completed work to be shared easily and remotely.
- Meistertask: Used to create story cards from tasks assigned to all group members, in tracking the weekly sprint.
- Mural: Allows remote collaboration via an online drawing/planning tool. Allowed the team to create storyboards and share ideas throughout the week.

A few weeks into the project, however, saw the disuse of both Meistertask and Mural; it was evident that there were too many platforms for team members to record their ideas and view details of current or upcoming tasks. Following similar feedback from the team, task details and storyboards were moved to Google Drive, leaving only two or three places that team members need keep up to date with. This proved more effective, allowing the team to focus on development rather than admin. In order to encourage regular communication throughout the team, the project manager routinely wrote into the group chat and paid attention to the replies; should any member not be responsive throughout the week, they would receive a personal message inquiring about their progress.

As regards communication with module staff, the project manager and lead programmer attended weekly meetings with the group supervisor, to keep him updated on progress or issues for that week. The module leader was consulted on a few occasions, regarding setting up a git server for the team, and further consultation on issues.

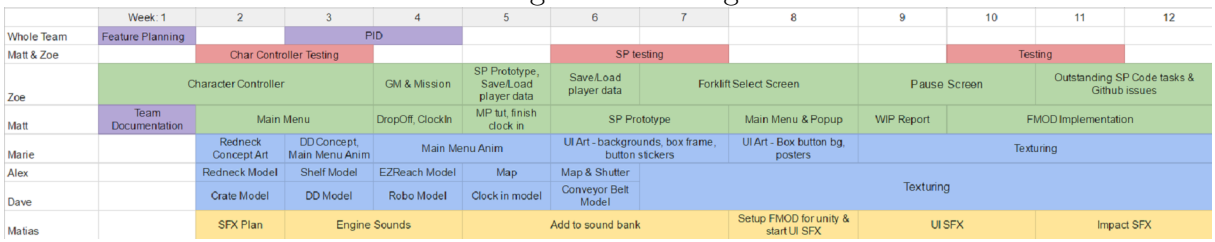
4.3 Schedule

Figure 10: Original Timeline Plan



The above figure illustrates the original gantt chart included in the PID, as was planned prior to project kick-off.

Figure 11: Living Timeline



	Week: 1	2	3	4	5	6	7	8	9	10	11	12
Whole Team	Feature Planning		PID									
Matt & Zoe		Char Controller Testing				SP testing				Testing		
Zoe	Character Controller			GM & Mission	SP Prototype, Save/Load player data	Save/Load player data	Forklift Select Screen		Pause Screen		Outstanding SP Code tasks & Github issues	
Matt	Team Documentation	Main Menu		DropOff, ClockIn	MP tut, finish clock in	SP Prototype		Main Menu & Popup	WIP Report	FMOD Implementation		
Marie		Redneck Concept Art	DD Concept, Main Menu Anim	Main Menu Anim		UI Art - backgrounds, box frame, button stickers		UI Art - Box button bg, posters	Texturing			
Alex		Redneck Model	Shelf Model	EZReach Model	Map	Map & Shutter	Texturing					
Dave		Crate Model	DD Model	Robo Model	Clock in model	Conveyor Belt Model						
Matias		SFX Plan	Engine Sounds		Add to sound bank			Setup FMOD for unity & start UI SFX	UI SFX		Impact SFX	

The above figure displays the actual timeline, which was kept updated throughout the project.

As is evident by comparing the above images of the teams predicted and actual progress, changes were made throughout the project. These changes were a necessity brought about largely by the difference between the projected and resulting time requirements of tasks. In keeping with Agile methodology, more time was allocated where necessary to certain tasks, thereby pushing back others. The original plan was useful in keeping goals in mind, whilst allowing a fluidity in the arrangement of tasks.

5 Conclusion

6 Appendix A

Figure 12: Project Initiation Document

Forklift Frenzy Video Game
Project Initiation Document

Edinburgh Napier University

SOC09109

40182161

40283659

40172073

40181382

40286188

Table of Contents

1 Project Goals

This group project module provides the perfect opportunity to allow a group of students keen to work on games in their future careers to create a portfolio piece, whilst gaining some insight and experience in working in a small team to publish a game created from scratch. As such, the aim of this project will be to develop a game for use on Windows PC, suitable for an audience of all ages, to be released on Steam Greenlight as a public demo.

The main goals involved include:

- Establishing a production pipeline appropriate for this type of software project, by means of creating and adhering to workflows that enable smooth interactions between the different disciplines of the team.
- Ensuring work done by all team members is standardised, and finished to a polished state considered suitable for public release. This includes thorough testing and evaluation of each piece of work.
- Arrange funds necessary for project success, to cover such things as server & Steam submission fees.

The game itself, named Forklift Frenzy, lets the player take command of a number of forklifts with which they must navigate the map to retrieve certain packages against the clock. A single player mode provides this gameplay offline, whilst an online multiplayer mode puts players in teams, with the friendly team collaborating together to gain as many points as possible by retrieving packages whilst the enemy team competes to fulfil the same task.

2 Approach

2.1 Methodology

2.1.1 MoSCoW

The MoSCoW method groups all project features into four categories; must have, should have, could have and won't have. It will be relied upon to aid with determining the priority of certain tasks, and afford a clear view of the project goals. This is not an exhaustive list of features, but a higher level view of what is required for the project.

2.1.2 Agile Software Development Methodologies

Certain characteristics of the agile methodology will be adopted throughout the project –

- Close communication and collaboration between all team members throughout, to ensure any problems that arise are dealt with swiftly and with greater efficiency
- Iterative development by means of working in weekly sprints, with clear goals per sprint, including stand-ups and retrospectives at each weekly meeting come sprint end
- Ensuring flexibility by adjusting to changes in task deadlines and requirements

By following this agile approach, the group will ensure there is always a working and up to date product relative to the current project deadlines and requirements. This will be versioned through the use of a master Git branch, in which only production ready code will be pushed. Every new feature will be developed on separate branches and merged into the master project when finalised.

2.1.3 Agile Client Communication

The project manager/programmer serves as the client for this project, allowing such benefits as direct communication with team members, more immediate decisions on any product changes and a technical understanding of the development process. This is beneficial as having a technically aware client means more flexibility on the overall project aims and goals. It is easier to understand and communicate to the client if the project scope becomes too large as they are directly involved in the development tasks.

2.2 Technology

2.2.1 Developer Environment

The project will be developed and built using the game engine Unity. Unity is an industry standard game development software that certain team members have experience with. Unity allows for the code base to be written in C#, using Visual Studio as the main IDE removes the need for the developers to learn an entirely new programming language or development environment. Unity is also useful as it comes with its own physics engine and rendering API which means that iterations of the project can be developed quickly and easily without the added difficulty of creating our own game engine.

2.2.2 Asset Creation

For the digital media assets the artist team will be using Auto Desk Maya 2016 to create the 3D models. Maya was chosen as the artists all have experience in using the software and the models can be imported into Unity with little effort. Adobe Creative Cloud will be used for other assets; including Photoshop for texturing and After Effects for video animation assets.

2.2.3 Version Control

Following industry standard software practises, the project will be stored using the version control system Git. This means that each team member can work collaboratively on the same Unity project by working on separate branches and issuing pull requests to the lead developer to merge and fix conflicts within the work. In using git, each commit of the project can be pushed to remote servers at GitHub.com and Bitbucket.org. The team are also researching into using university resources to set up a private git server so the large binary files can be stored. If a private git server is not available, GitHub will be used to version the code and google drive will be used to store the larger files. The GitHub web interface will also be used to log issues and create milestones for the project.

3 Scope

The main goal of this project is to develop an original game. The game is to be aimed at Windows PC users with possible Mac/Linux support. The game should be functional and ready to be released as an alpha development build, which will be uploaded to Steam Greenlight in the form of videos and screenshots of gameplay. The game should feature both offline and online modes: a single player repeatable free run mode; and an online multiplayer versus mode.

3.1 MoSCoW Format

The following are the deliverables of the game in the MoSCoW format:

3.1.1 Must Have

- Single Player mode
- Multiplayer mode
- Menu system
- Local Player Data Save/Load Capability (Game persistence)
- Server to host multiplayer games
- Four different forklift classes

3.1.2 Should Have

- Multiplayer: leader boards of top scores
- Controller support
- Multiplayer: matchmaking system
- Server database to store online player statistics
- Forklift damage system
- Point-based system
- Steam SDK integration for release on Steam platform
- Steam Greenlight submission (screenshots & gameplay video)

3.1.3 Could Have

- Multiplayer: leader board system (shows friends scores)
- Different types of missions in either/both game modes
- Game e-manual: includes concept art, descriptions and instructions
- Mac support
- Teaser for further levels & models
- User defined key bindings
- Steam achievements
- Unlockable character skins

3.1.4 Won’t Have

- More than one map
- Multiple game modes within single or multiplayer
- Micro-transactions

4 Organization

4.1 Management team structure

4.1.1 Team roles

Project Manager & Programmer – Matt Newbigging – responsible for managing documentation, organising tasks, ensuring the project is developed to specification and by the module deadline. Alongside this, perform some programming tasks.

Lead Developer – Zoe Wall – setting up version control method and upkeep, developing the majority of the code base required for the game.

Artists – Marie Pearson, Alex Haining, David MacGruer – creation of all game assets; models, animations, textures and video material.

Sound Designer – Matias Malmivaara – recording, mixing and implementation of all sound assets in game; background music/ambience and sound effects.

5 Business Case

5.1 Benefits

The approach chosen involves using the Unity game engine, since some members of the team have existing experience in using it, it comes with it's own physics engine and provides a simple interface for new learners whilst providing advanced functionality. This affords a major benefit to the project, in contrast to other approaches which would involve learning how to use a new game engine, or even building our own game engine from scratch, which would put this project out of scope from the offset.

Expected Outcomes:

Provide experience working on a published title – This project provides an opportunity unique to university by allowing the team to work on a game that will be made commercially available. As such, should the game be accepted to Steam, it will generate a small amount of income.

Fills a niche in the market – There exist forklift simulations which are very true to nature. There are not, however, any games that use this theme to bring something lighter in nature to the market – this is very much a silly game that does not take itself seriously.

Provide experience working with and managing different team member's skills and needs – This project provides an excellent opportunity for both artists and programmers to collaborate as if they were working within the industry. With every team member having to understand and appreciate one another's background by making sacrifices or changing the way they work to suit everyone.

6 Constraints

6.1 Project Limitations

The project will be limited with regards to the following:

- Time limit imposed by the module length means project must be complete by 03/04/2017
- Limited time available due to other module commitments during project period
- Developer teams current knowledge of Unity
- Financial requirements – whilst a grant may be available for this project, there are certain costs associated with publishing the game to Steam, and in server costs necessary for making the game available for online play
- Artists current knowledge of how their assets affect how the developers work
- Possible limitation due to Project Manager having a dual technical role, some aspects of development may be out of scope due to having a sole developer
- Work-flow changes for team members to be collaborative.

6.2 Iteration Tolerances

Factors to be considered during every project iteration:

- Possible risks
- Time constraints
- Scope of current tasks
- Quality controls

6.3 Assumptions

- Development team's knowledge of Unity will meet requirements of application
- Timeframe will be sufficient for completion of the project
- Team members will hit individual deadlines
- Work will be performed in accordance with agreed team rules

7 Stakeholders

As the project has an internal client, the team are the main stakeholders of this project. This is useful as overall project goals and features can be flexible to suit any unforeseen issues or technical requirements. One goal of this project is that the game be submitted to Steam Greenlight, therefore the end users are also stakeholders in the sense that the public must vote if they believe the game should be a released title, which could help with the redesign of the project if it proves to be unpopular.

8 Risks

Risk Identification						Qualative Risk Assessment		Risk Response Plan		Monitoring
Risk ID	Risk Category	Risk Event	Cause	Effect	Primary Objective	Probability	Impact	Response Strategy	Response Actions	Interval Check
1	Internal	Lost resource	No backup of files / Hardware failure	Further time spent on repeated work, deadlines not met	Time	Low	Low	Mitigate	Recover document, use similar document or repeat work, back up work to multiple servers, roll back to previous commit	Weekly
2	Project Management	Absent team members	Sickness, adverse weather, emergency	Hinders progress, deadlines not met	Time	Medium	Medium	Transfer	Other team members to absolve tasks of absentee where possible, change meeting times	Daily
3	Project Management	Teamwork issues	Personality clash, wrong project decisions	Team communication & morale suffers, hinders progress	Time	Low	Medium	Reduce	Agree on ground rules and product decisions	Weekly
4	Technology	Lack of skills	No previous/sufficient knowledge to complete task	Hinders progress, deadlines not met	Time	Medium	Medium	Transfer	Individual should self-teach in order to complete task to standard	Weekly
5	Project Management	Lack of funds	Unable to secure a grant	Unable to afford server & Steam submission fees	Cost	Medium	High	Transfer	Team members to front cost	Following grant application
6	Project Management	Unclear communication	No regular team meetings, no comms channels set up	Incorrect project direction, hinders progress	Time	Low	High	Avoid	Hold weekly meetings and ensure communcations channels are viewed daily	Daily
7	Internal	Lack of consistency	Artists don't adhere to same standards	Assets don't look similar, have disimilar properties	Time	Medium	Medium	Avoid	Agree on asset standardisation rules	Weekly
8	Project Management	Scope of features too high	Too many desired features, additional necessary features are discovered	Unable to finish project to standard	Time	Medium	High	Mitigate	Only work on features essential agreed by the MoSCoW standard, keep up to date. Remove features following Agile Methodolgy	Weekly

9 Project Control

9.1 Team Ground Rules

- Check Slack regularly, be responsive
- Task review in each weekly meeting
- Give notice of absence from meetings
- Make PM aware of anything that would disturb progress on weekly tasks
- Master branch in version control must be production ready at all times

9.2 Team Communication plan

Group meetings to take place every Tuesday between 11:00 and 13:00, wherein:

- Team members perform a stand-up; detailing their progress on their most recent task, any problems faced and any questions they need ask.
- New tasks with associated deadlines assigned to team members where necessary.
- Minutes taken by PM to appear in Group Meetings document on Google Drive.

The team will make use of Slack to allow for easy communication between individual team members and as a group. Google Drive will be used to store all assets created for the project, including concept art, models, sound files etc. Mural will provide a means of illustrating storyboards, final designs and brainstorming remotely. Project version control will be achieved through the use of Git. The project will be pushed to a remote repository on the GitHub server.

9.3 Communication Summary

- Project manager to arrange weekly group meetings.
- Project manager to check in with supervisor and module leader where necessary.
- Project manager to present the assessment documents to the module leader.
- Project manager to meet with other project managers weekly at the project manager's forum.
- Project team will show a project presentation to module leader at the end of the project.

9.4 Maintenance and Control Documents

Issue Log – Used to track issues on GitHub when they arise during the project. Team members should create issues using the GitHub interface with as much detail as possible for things that need to be rectified at a later date. Not exclusively used for bugs when testing the game, the issues can be to further improve a feature. This can be used to keep track of features needing extra work when the time allows later in the project. Once an issue has been opened, any member of the development team can then include the issue ID within the commit that fixes it to keep track of the iterative improvement process. Any issues that arise that are not related to bugs or improvements to the game should be communicated through Slack, if a suitable solution can be found immediately it will be carried out by the team member. If a solution cannot be found immediately, the issue will be discussed in the following group meeting.

Risk Register – Used to log all possible risks during the project along with the probability of occurrence and their impact on the project should they occur. This will be updated as new risks become apparent. This document will be controlled by the project manager.

Git Commit Statements – These are short statements which go alongside a commit to the version control repository. Each statement provides a description of the changes made by the commit and any further work to be carried out. They are written by the individual team developer making the commit. These are to be descriptive and contain the Issue or Milestone ID if the change is relevant to an open case on GitHub.

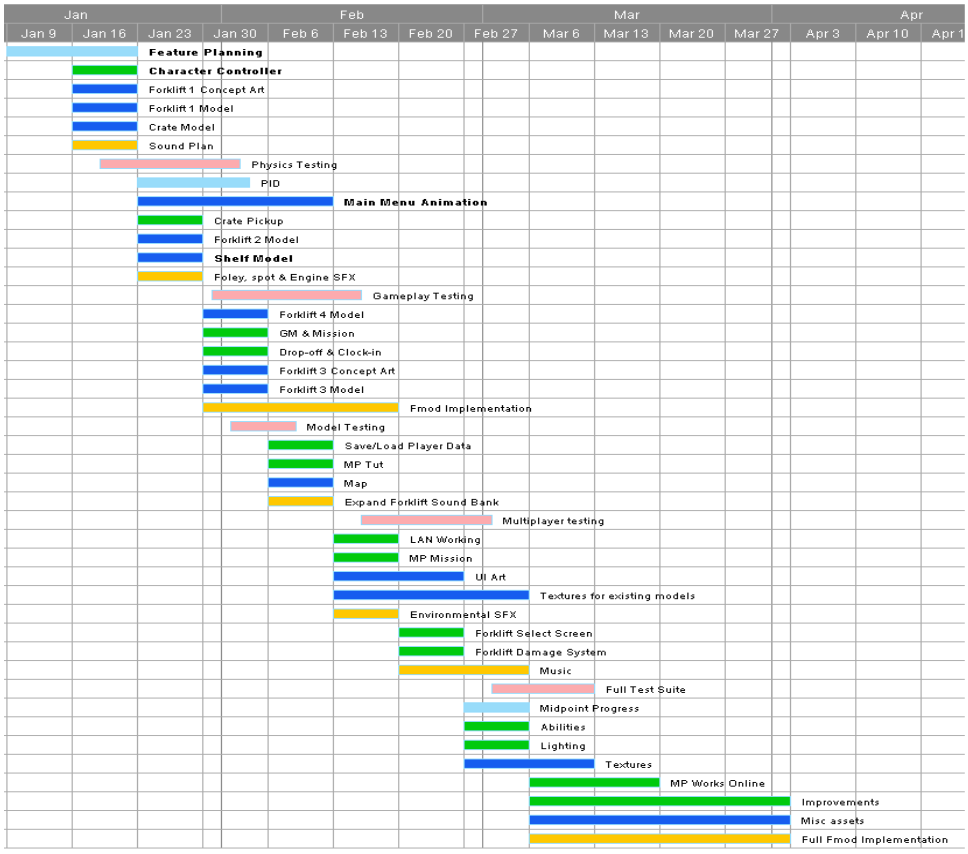
10 Reporting Frameworks

The following are all forms of documentation required to be produced during the lifecycle of the project.

Report	Recipient	Date Expected
Project Initiation Document	Module Leader	03/02/2017
Work-in-progress	Module Leader	10/03/2017
Project Report	Module Leader	03/04/2017
Project Presentation	Module Leader	24/05/2017
Individual Reflective Report	Module Leader	03/04/2017
Individual Project Diary	Module Leader	03/04/2017

11 Schedule

The following Gantt chart shows an outline for the schedule of the project.



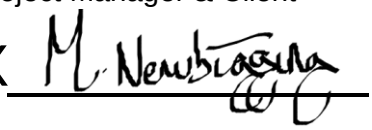
12 Sign Off

I confirm that this project initiation document accurately describes the scope of the work proposed as of the current date and ask for the project to commence.

Any changes to the scope above will be subject to change control. This project initiation document is valid for five months after issue.

1st Authorising signatory

Project Manager & Client

X 

Date - 03/02/2017

13 Appendices

13.1 Appendix A: Personal Learning Outcomes

13.1.1 Alex Haining

I hope to take my skills from my 3D module last trimester and develop my talents further in my group, by creating a game in the unity engine. My contribution within the group is valuable and will push my skills forward by producing optimized game ready 3D assets for my team. I have been given the task of creating assets partly from my own creativity and following 2D concept drawings. I will broaden my abilities in Autodesk Maya and improve my team workflow by understanding production pipeline involved.

Main learning outcomes:

- Create Low poly assets.
- Optimize assets to game standards.
- Learning how to export 3D Maya files into the unity game engine.
- Working close and collaborating with other artist to ensure the work produced is within the accepted same standard and style.
- Learning basic mechanics of the Unity Game engine allowing me to integrate the modelled assets before handing over.
- Creating Normal maps and hand painting via photoshop to create the texturing in the desired style.
- Learn and use the basic understanding of GIT Hub to transfer files between the Game developers

- Giving a full professional breakdown and evaluation of my work that can be presented towards the client.

13.1.2 David MacGruer

Following on from trimester one where I studied the 'Introduction to 3d modelling' module, I plan to utilize many of the skills I learned to produce game ready assets for integration into the Unity game development engine.

I have been tasked with producing a range of 3d assets that will populate the game, bringing 2d concept drawings to life in the 3d realm.

Using Autodesk Maya 2016 to model the assets, I will expand my knowledge of the game production pipeline.

Main Learning Outcomes:

These outcomes will be achieved through self-directed study.

- Low polygon modelling techniques for optimum performance in game.
- Creating hand painted texture maps for the 3d assets using Adobe Photoshop.
- Working closely with other artists and game developers to ensure my work is presented to them in a professional fashion to avoid any glitches with regard best practice for integration with the Unity game development engine.
- To attain a basic understanding of Unity, to enable me to test my 3d assets within the game engine before sending them on to the Lead Developer.
- Take part in a weekly meeting with my group to discuss progress on the project and any outstanding tasks to complete for each production stage.
- Complete a detailed breakdown of all the game assets I have produced including a critical evaluation of my work for consideration by the client.

13.1.3 Matias Nilla Matias Rafael Malmivaara - 40286188

This group project gives me the opportunity to further advance my knowledge and ability in interactive audio and sound design for video games. I'm required to create all audio assets to the game myself, as well as the music, so there is a whole aspect of the game depending on me. I've been given almost full creative control over the audio and sound design, which should be both challenging and rewarding, though the setting of the game is rather specific and thus gives helpful boundaries to work within. I hope to be able to use what I already know, push myself a bit further, and learn a whole lot of new things that will help me professionally in the future. The knowledge gained in modules last trimester, such as "interactive audio", is directly linked to this line of work and will be extremely helpful. The project will also give me an opportunity to further familiarize myself with game development related audio software / middleware.

Main learning outcomes:

- Learn about and gain basic understanding of coding based game development, specifically in relation to audio

- Further develop skills in manipulation of sound through the use of recording, layering, and processing the audio, using appropriate microphone techniques and tools such as Pro Tools.
- Get better at data organisation, file organisation, asset organisation
- Further develop understanding and features of audio middleware, in this case FMOD Studio
- Develop further skills in implementation of audio in video games
- Be able to create a balanced and dynamic sound mix for a game
- Further develop in music implementation in games
- Learn to use and understand the basics of GitHub
- Learn the basics of a new game engine (Unity), specifically audio implementation
- Develop group working and communicational skills, being able to communicate and develop ideas from others and myself to a unified standard
- Provide a full breakdown of the audio assets and sound design elements I've created for the project
- Critical evaluation of my own work

The outcomes will be achieved through self-directed study and collaboration with other group members.

13.1.4 Matthew Newbigging

I aim to improve upon existing project management skills through working with team members of different disciplines to my own, understanding the production pipeline in developing a game, and ensuring work is completed to specification by the deadline . Furthermore, I plan to better my knowledge of Unity, Git and code structure via my programming contribution to the project.

Main Learning Outcomes:

- Create accurate, readable & easily accessible documentation for the project.
- Able to successfully manage the team's schedule throughout the module.
- Use Git effectively, to safely work simultaneously with others and provide a means to access my work remotely.
- Write various code files for the game, to specification.
- Better knowledge of using Unity to develop publishable games.
- Understand pipeline and all requirements involved in developing a game across a multi-disciplined team.

13.1.5 Marie Pearson

I intend to develop my existing skills further within the game creation project through working within the team as a Concept Designer and 2D Animator, producing pieces that will aid the 3D artists in the production of in game assets and also to produce an opening 2D animation sequence which shall contribute to the overall character of the game. By building upon

knowledge gained from previous modules such as 2D animation and Introduction to 3D, I will gain experience of integrating concept design into the production workflow and enhance my team working abilities through cooperation and understanding of the overall creation process of the project.

Main Learning Outcomes:

- Utilise existing skills in Concept Design and refining skills to a greater degree
- Creating designs and art work that may be utilised in game and by fellow artists.
- Gaining a basic understanding the Unity engine in relation to my role
- Contributing to team meetings through discussion and implementing ideas into concept designs
- Working within the team to ensure that concept ideas meet the overall expectations of the project, particularly in relation to design implementation by other team members, providing additional art work when required.
- Producing a short 2D animation sequence for initial menu through use of Adobe CC
- Providing a portfolio breakdown of Concept Designs and rational for presenting to the client

13.1.6 Zoe Wall - 40182161

During this project I intend to build on previous knowledge of Unity. In particular learn more about its physics engine to create interesting physics simulations of vehicles, crates affected by force and how to apply this to models effectively without hindering application performance.

Having no prior knowledge into how online games work, this project is an opportunity to research and implement simple networked code, crucial for the multiplayer aspect to games development. I would like to learn how to develop a game where at least two users can successfully interact within the same level in real-time.

I also wish to gain a better understanding of Git by fixing merge conflicts and creating branches to follow a standardised git-flow used within the industry. Previously I have only ever used version control for solo projects, so this project is a perfect opportunity to learn how to effectively use git to work collaboratively with others.

Main learning outcomes for the project:

- Develop further understanding of version control and its proper use in a collaborative project: branching, pull requests, merging, fixing conflicts.
- Find a solution for large file storage for the binary assets that may need to be versioned.
- Gain a more in depth knowledge of the Unity engine with regards to the physics engine: rigid bodies and accurate collision detection.
- Learn how to integrate data serialisation to create a game which is data persistent between executions.
- Build on prior knowledge of game development patterns and techniques to create a fully functional game with interesting mechanics.
- Working closely with artists to further an understanding of asset creation and how to integrate it correctly with the development work.
- Learn and implement the basics of networking for online multiplayer games: servers, clients and hosts.

7 Appendix B

Gantt charts

8 Appendix C

Test results:

9 Appendix D

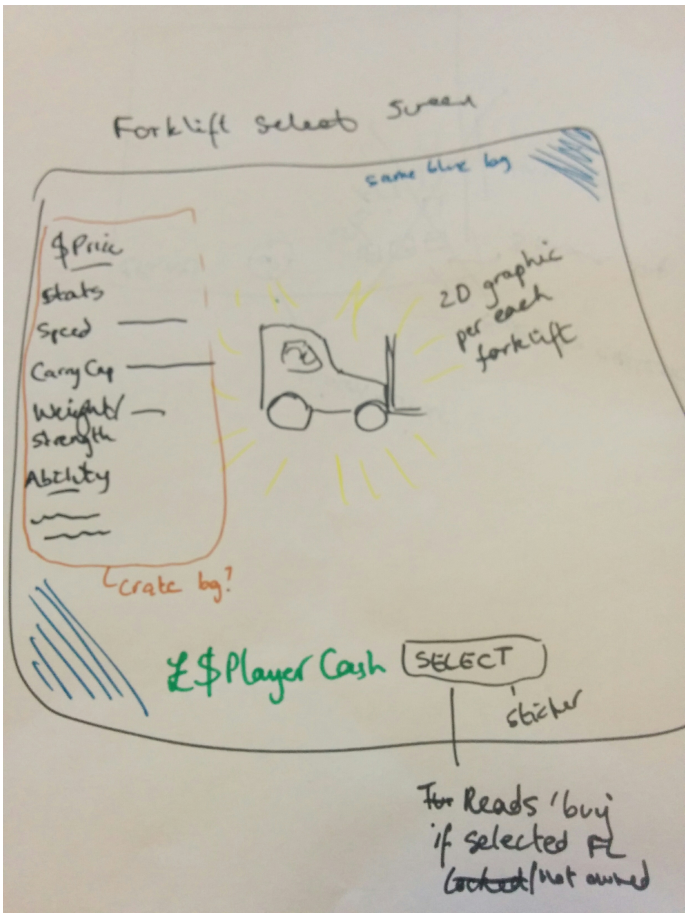


Figure 13: **Pre-production** - concept design for forklift selection scene.

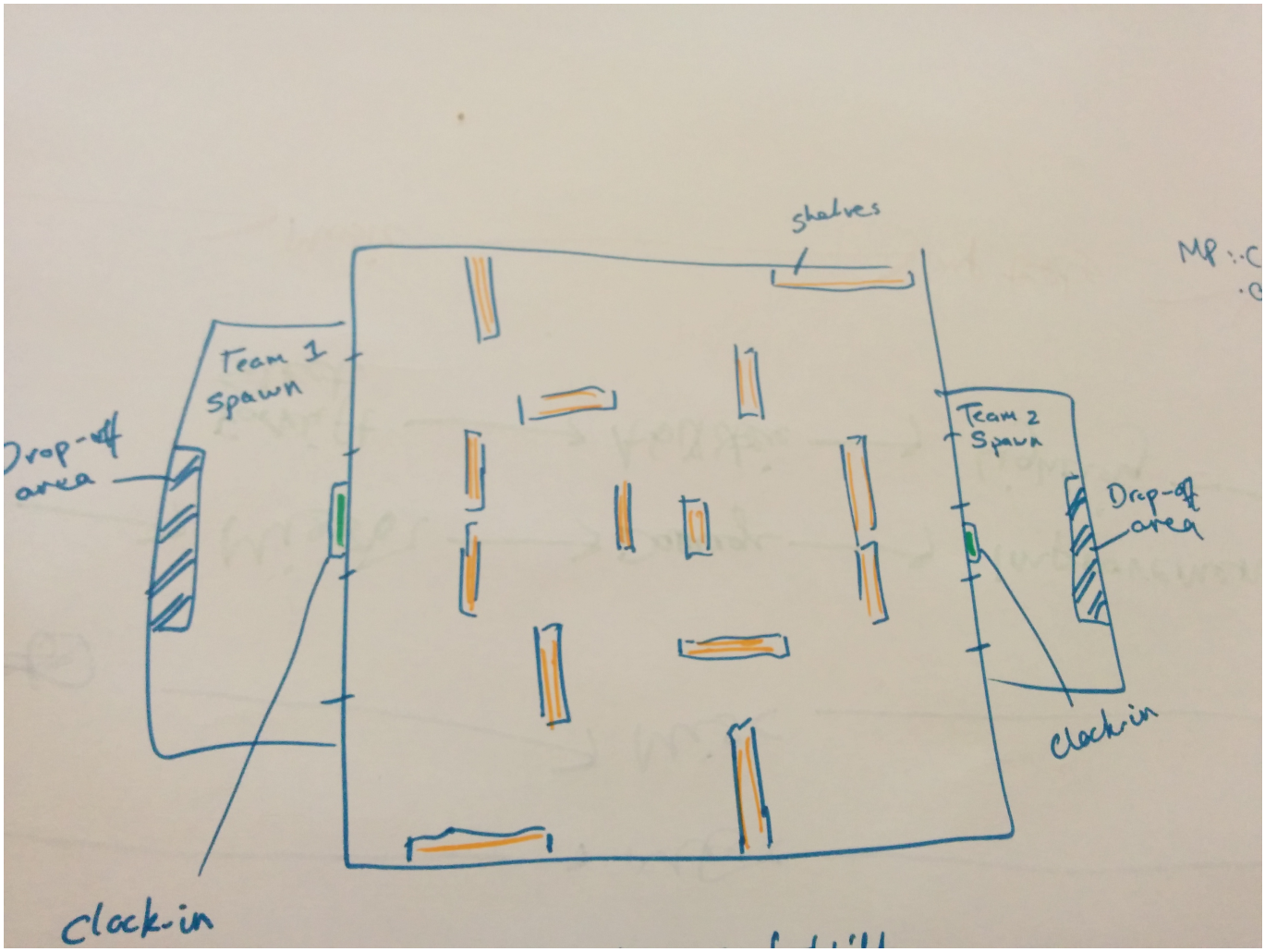


Figure 14: **Pre-production** - original multiplayer map design.

Figure 15: **Pre-production** - Clock-in area concept.

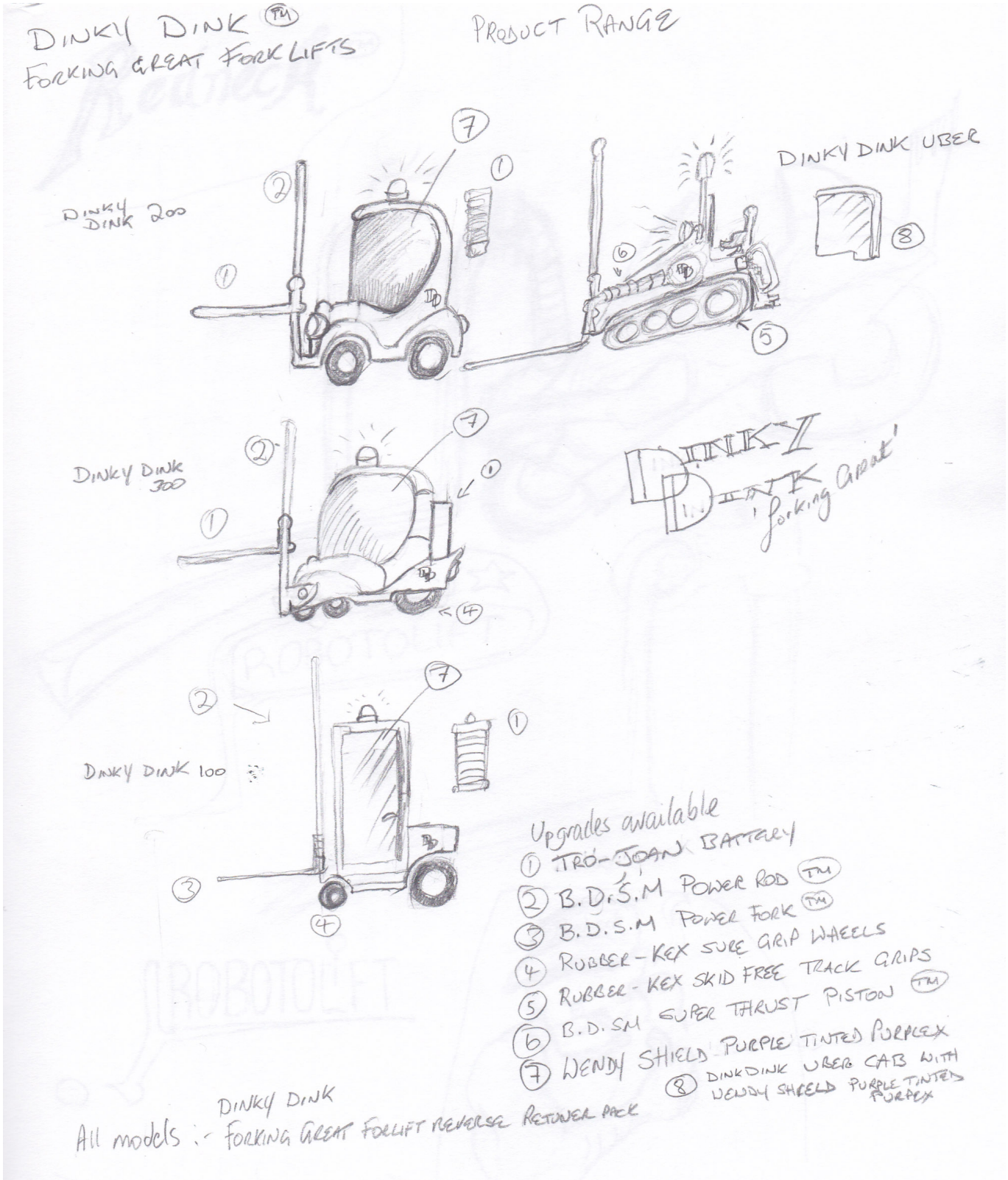


Figure 16: **Pre-production** - concept drawing of a forklift.

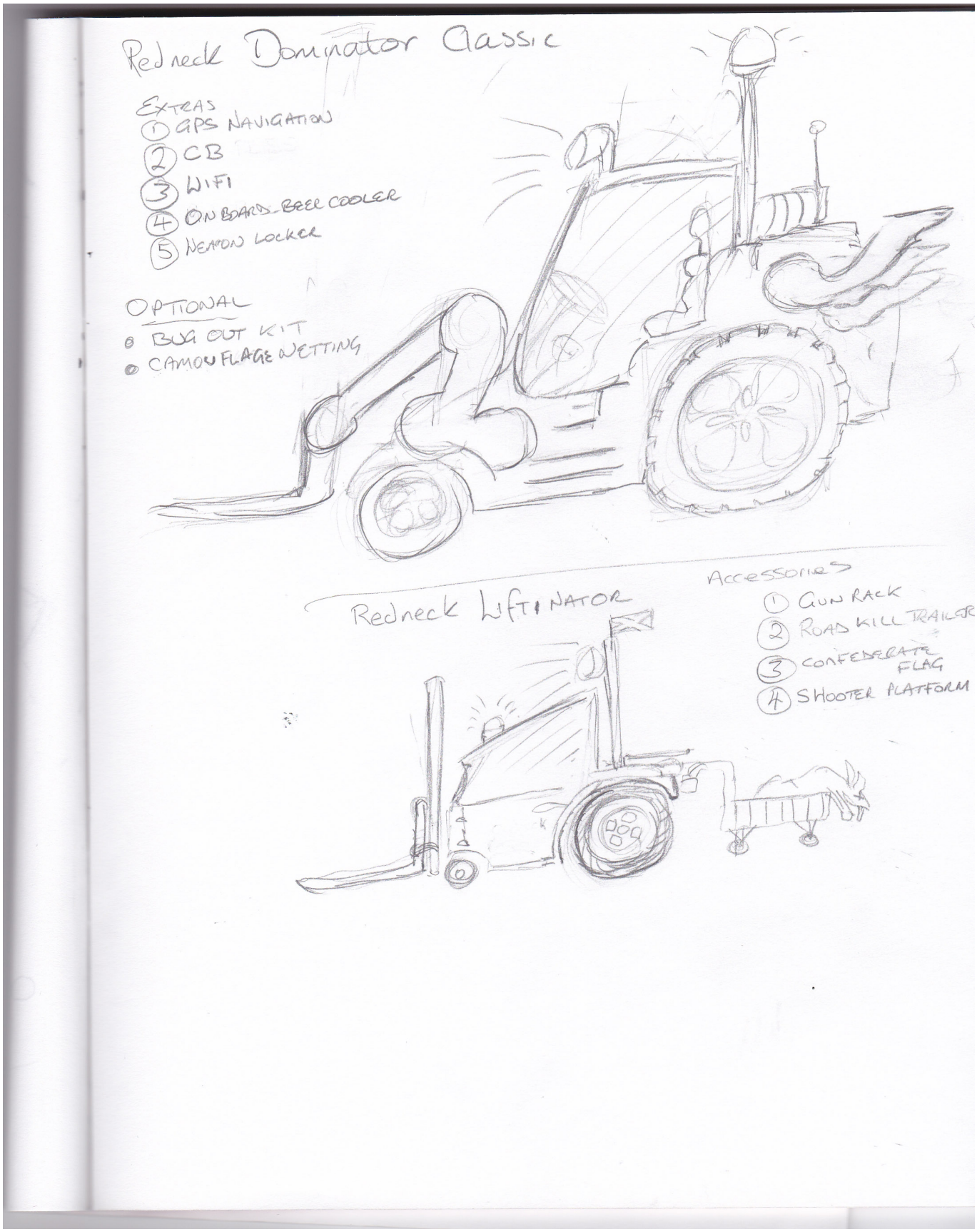


Figure 17: **Pre-production** - concept drawing of a forklift.

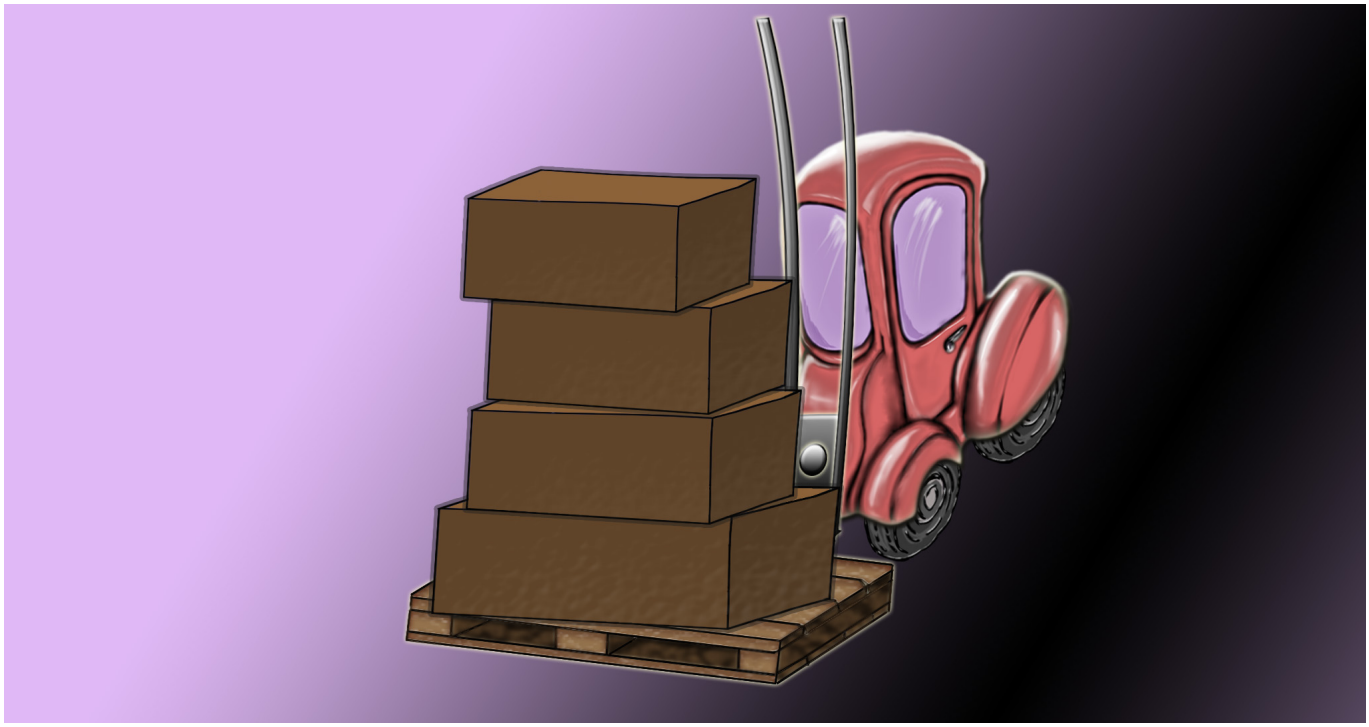


Figure 18: **Production** - animation screen capture of main menu.

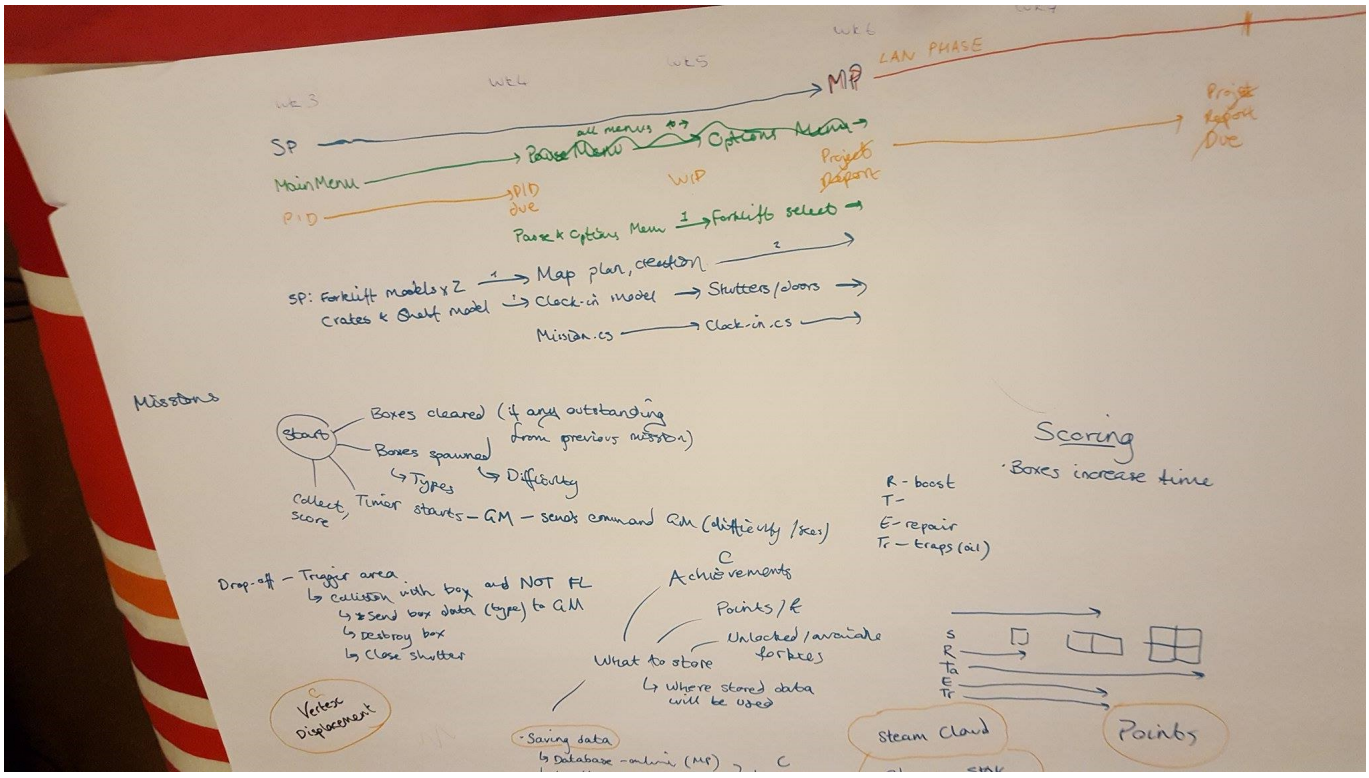


Figure 19: **Pre-production** - code achievement and scoring design.

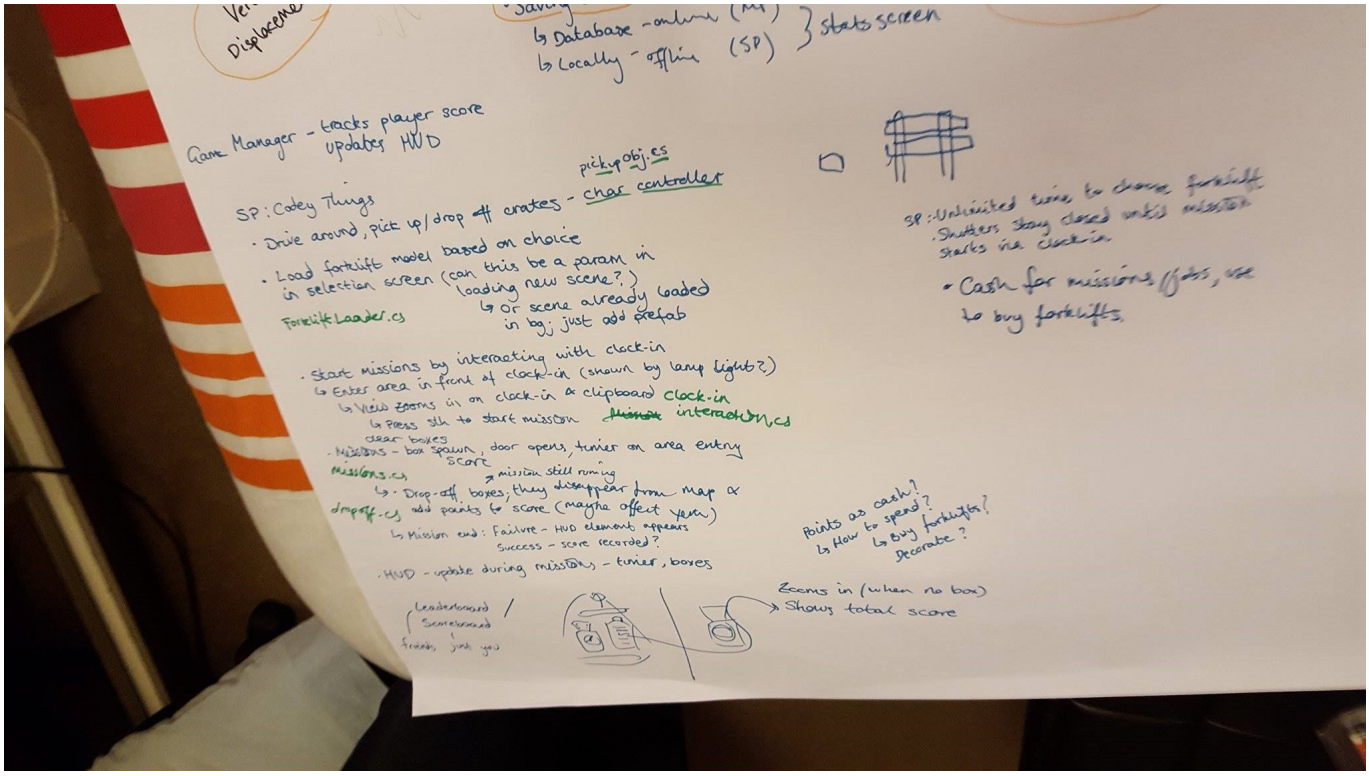


Figure 20: **Pre-production** - code layout and design.

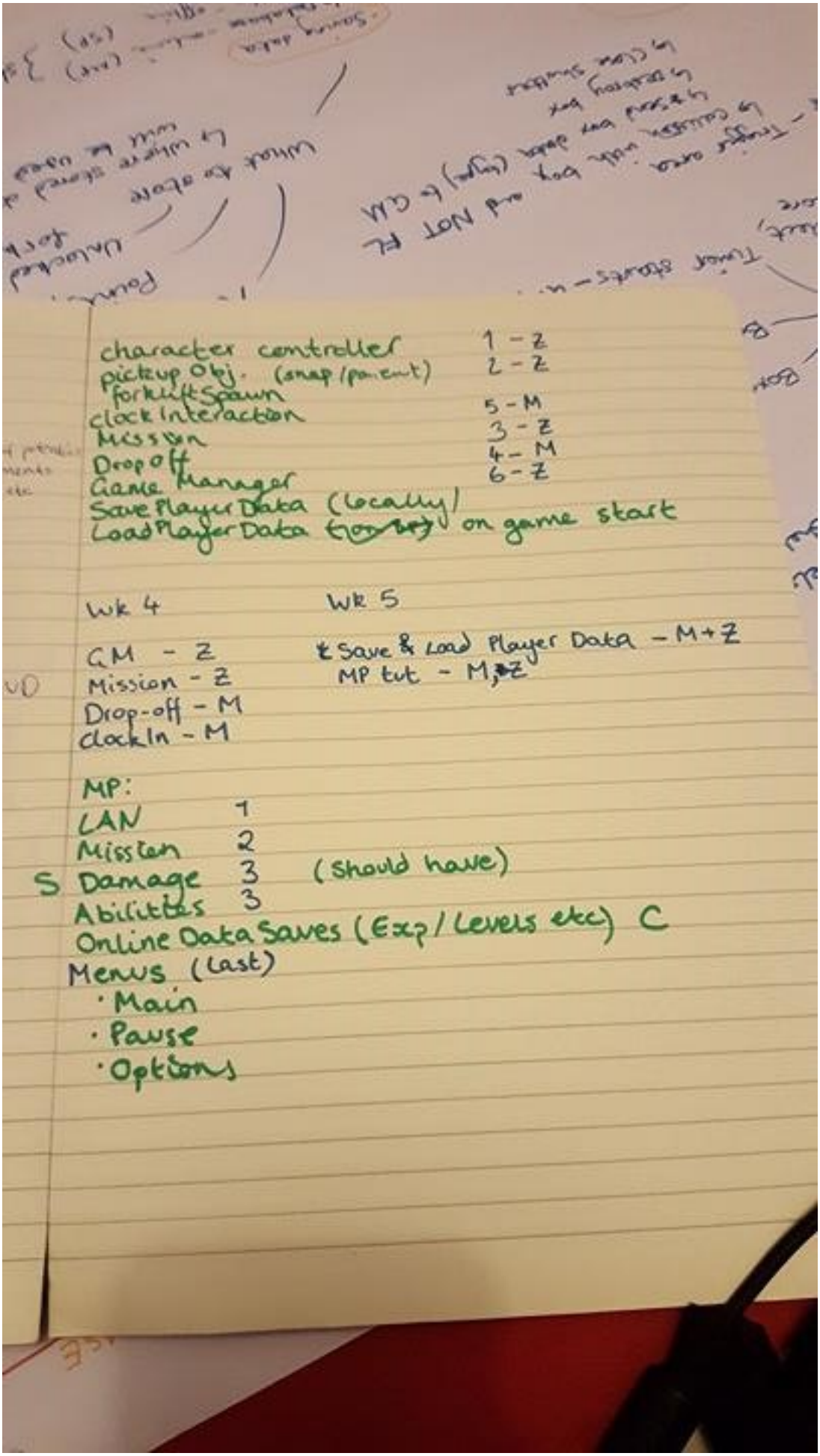


Figure 21: **Pre-production** - code task distribution.

Figure 22: Code definitions

03/05/2017

Game Manager

- StartMission(difficulty)
  - Called from Clock-in
- Deliver(box)
  - Called from Drop-Off
  - Add to player points based on box type
  - If special box, AddTime(value)
- StartTimer(difficulty)

- PlayerScore
- Timer

Mission

- ClearBoxes
  - From previous mission, if any left over
- SpawnBoxes(difficulty)
  - Box Count
  - Random box spawn locations
- GM.StartTimer(difficulty)
  - OnTimerEnd
- OnMissionSuccess - all boxes have been dropped off

Drop-off Area

Waist-high conveyor belt platform. L shaped to fit in corner of spawn area. Large enough to be approached by 3 forklifts.

- Belt always moving; only boxes placed
- Destroy box in an area (once it's gone 'outside' of map, & can't be seen)

Clock-in

- Enter area - zoom in to clock in
- Select difficulty
- Press start
  - Send difficulty to GM - GM.StartMisison(difficulty)
- Exit area (driving as normal) - zoom out again

1

Command pattern

Can reassign buttons using this for controller or mouse/keyboard support.

Single Player, Multiplayer scenes

User Controls

- Drive
- Reverse
- Turn left/right
  - Animation for wheel turning (regardless whether player has stopped)
  - Let go of turn key, wheels return to forward position
- Lower/Raise forks
- Ability
- Camera rotation

Control scheme - which keys make sense to use?

Scripts needed

- Abstract base Command
- InputHandler to return Command (Attached to player)
- Each command concrete imp takes in actor parameter
- Moving - when button pressed for move forward; translate by drivingAngle property.
- When turning left/right; alter drivingAngle

Picking up/Dropping off Boxes

- Use pallets - boxes on top of pallets, to allow forklifts to pick up
  - Are the boxes fixed? Can they be dropped from the pallett?
- Boxes magically snap to forks?

Main Menu

Single Player	- Loads single player scene, which prompts forklift selection scene
Multiplayer	- Loads multiplayer scene, prompts select scene
Options	- Shows options screen, allows input configuration
Exit	- Shows Are You Sure? Prompt, quits on Yes.

Main Menu > Options

- Allow choice between keyboard/controller input
  - Button mapping for both?
- Audio options
  - Volume
- Graphics options
  - Any pp effects?!
  - Resolution?

Figure 23: Sound implementation details

Event	How it should work
Engine Full	Better option: Based on forklifts speed (variable?): RPM parameter 1k-8k If throttle held down: load parameter 2000 (plays onload), if not held down: load parameter -2000 (plays offload) Damage parameter 0-100 based on damage in game. Alternative for speed: If throttle held down: RPM parameter changes immediately to 8k If throttle not held down: RPM parameter changes immediately to 1k *The alternative option needs simple alteration in FMOD, so let me know if this is the one you'll end up using*
Forks_up	Forks up button held down: Play event (not looping!)
Forks_down	Forks down button held down: Play event (not looping!)
Forks_Impact_ShelfMetal	When forks collide with metal material: Play event (not looping!) *Event should to be played at the impact location* *Would be optimal to change volume according to impact strength.*
WoodCrateBreak	When a crate breaks on impact: Play Event *Would be optimal if impact strength would determine parameter "ImpactStrenght" to be 1 if low, and 2 if high.* *Alternatively can use volume adjustment based on impact strength same as with metal impact sound.*
Clock_Card	Whatever logic in game for using the clock card machine, so e.g. when clicking on a mission: Play event (not looping!)

Conveyor_Belt	Place to the location of the belt. Make sure it's on "auto play" so that it starts playing when level starts. I will see attenuation settings when I get to test it.
WoodCrate_Normal_Impact	When non-destructive collision with a crate: Play event. *Again, either volume based on impact strength, or a parameter based on impact strength as with crate break.* *Should be triggered at the impact location*
Air_Conditioning	Place to the location of Ventilation shafts /aircon, whatever it might be. Make sure it's autoplayed when level starts. This will loop forever. I will see attenuation settings when I get to test it.
Mouse_Click	Play when clicking a menu item *Currently I've added a woosh as a tail, which I think sounds nicer, but might be odd for menu changes like vsync on/off, key binds etc, so let's look at it later again.*
Mouse_Hover	Play when hovering over a menu item
Engine_2_Full	Same as the other engine
Forks_Up_Tail	When let go of fork up button, play event. (will be a quick sound for tail, so it doesn't just end abruptly)
Forks_Down_Tail	When let go of fork up button, play event. (will be a quick sound for tail, so it doesn't just end abruptly)
TimeIsRunningOut	Should Start playing exactly when level time reaches the last 25 seconds…
Opening_Menu (under music)	Start playing when in menu (parameter "in menu" 1=yes 0=no)