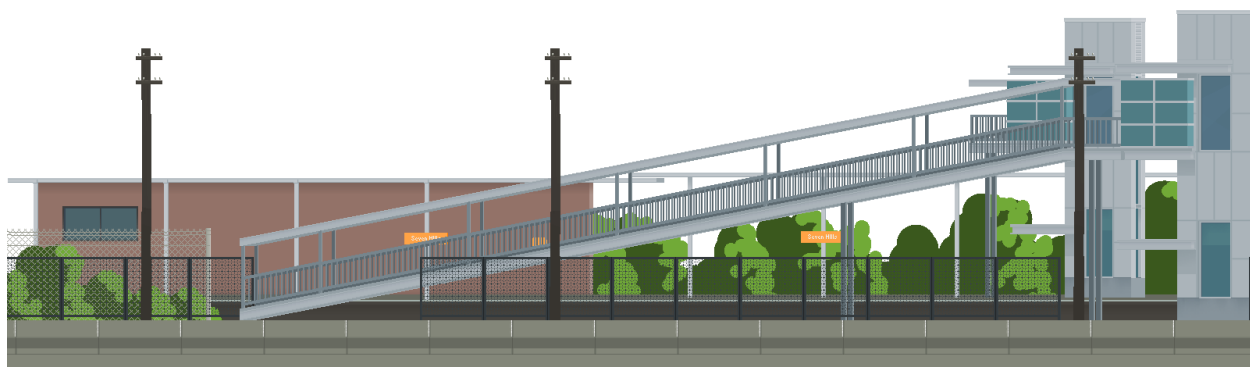


The Direct Daylight Project



Alex Tan
October 5, 2019

Contents

1	Introduction	1
1.1	The Team	1
1.1.1	Alex Tan	1
1.1.2	Brian Huang	1
1.1.3	Dylan Kroft	1
1.1.4	Xing Lin	1
1.2	Open Positions	1
1.3	Weekly Meetings	2
2	Documentation	3
2.1	Google Drive	3
2.2	Trello	3
3	SSH & Git	5
3.1	Generating an SSH Key	5
3.2	Setting Up Git	5
3.3	Command Prompt Basics	6
3.4	Cloning the Repo	6
3.5	Git Primer	6
3.5.1	Committing Files	6
3.5.2	Pushing Your Own Changes	8
3.5.3	Pulling Other People's Changes	8
3.5.4	Merge Conflicts	9
3.5.5	Stashing	9
3.5.6	Viewing the Project History	9
4	Code	10
4.1	Eclipse	10
4.1.1	Setting Up Eclipse	10
4.1.2	Running the Game	10

1 Introduction

The Direct Daylight Project is a 2.5D, retro-style, story-driven game, with its key themes centred around nostalgia and discovery.

1.1 The Team

1.1.1 Alex Tan

Role: Technical Lead

Google Drive: `torc.madra@hotmail.com`

Trello: @alextan60

1.1.2 Brian Huang

Role: Musical Lead

Google Drive: `brianhuang324@gmail.com`

Trello:

1.1.3 Dylan Kroft

Role: Graphic Designer

Google Drive: `dylankroft@gmail.com`

Trello: @dylankroft

1.1.4 Xing Lin

Role: Artistic Lead

Google Drive: `elevatedcross.ing@gmail.com`

Trello: @xinglin7

1.2 Open Positions

1. Artist

- for drawing people

2. Scriptwriter

- for fleshing out and writing the dialogue of the story

3. Voice Actors

- after the script has been finalised
- if possible

1.3 Weekly Meetings

There will be weekly meetings held in-person to discuss the progress of the project. These will generally be held on Sunday afternoons. Contact Alex for more information.

2 Documentation

2.1 Google Drive

The majority of the project documentation can be found in the shared Google Drive folder. Contact Alex for more information.

The main document *Direct Daylight* contains detailed information about the game including the overall themes, mechanics, gameplay, storyline, progression, and the implementation technicalities.

The *Artwork* document is used for documenting details about art in the game.

2.2 Trello

Trello is an app used for organising and planning out tasks, which helps in tracking the progress of the project.

- Sign up for Trello at <https://trello.com>
- Accept the Trello board invitation at <https://trello.com/invite/b/CW6IRbz0/b1605676bf57e5327f3cd156d03979d7/direct-daylight-game>
- The Trello Board can now be edited at <https://trello.com/b/CW6IRbz0>

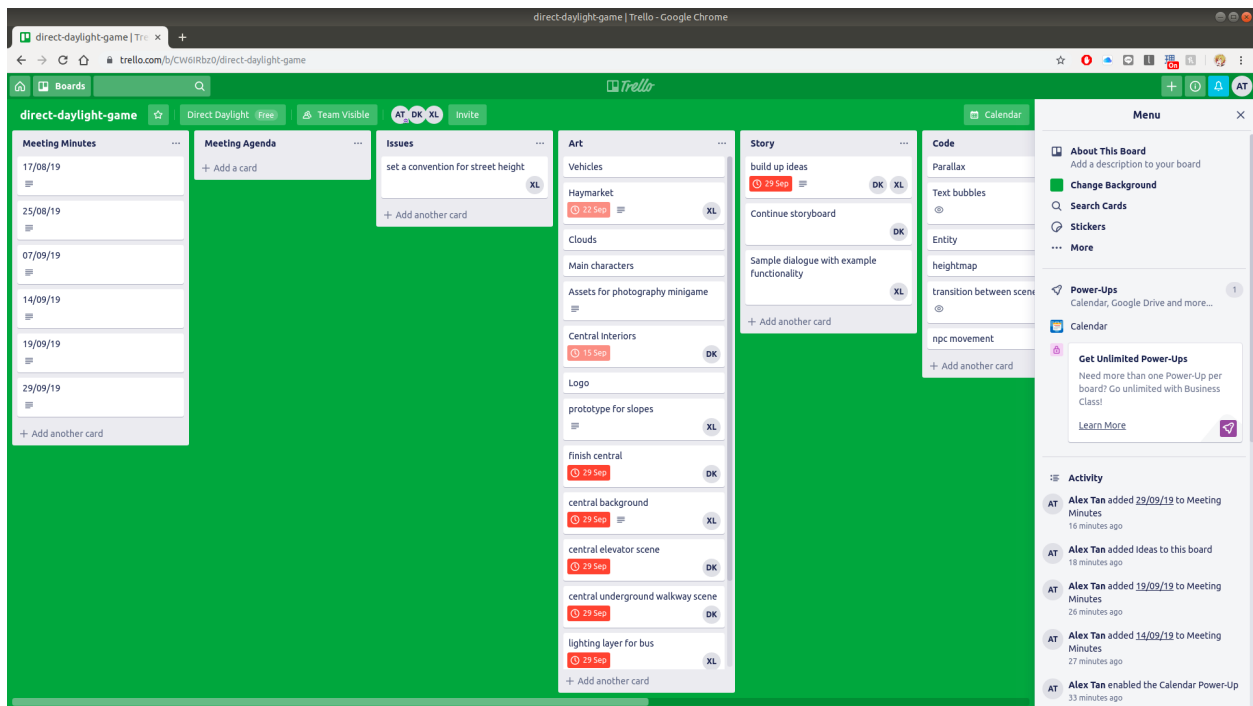


Figure 1: *Direct Daylight* Trello Board

Please try to stick to the schedule and meet deadlines so that the project can progress swiftly. Make sure you check the Trello board before each meeting to ensure you have completed the assigned tasks.

Previous meeting minutes, as well as the agenda for the next upcoming meeting will also be on Trello.

3 SSH & Git

This section covers how to download the project files, make changes, and sync these changes so that it can be viewed by the whole team.

3.1 Generating an SSH Key

You will first need an SSH key to gain access to the project files. Think of an SSH key as like a special password. To get started, follow the following instructions.

1. Open a terminal.
2. Run `ssh-keygen`.
 - You will be prompted for a location to save the `id_rsa.pub` file. Leave this blank to save it in the default location.
3. Take note of where the `id_rsa.pub` file is stored. You can find the location of the file in the command prompt.
 - On Windows, this is usually `C:\Users\your_username\.ssh/id_rsa.pub`
 - On Mac, this is usually `/Users/your_username/.ssh/id_rsa.pub`
 - On Linux, this is usually `~/.ssh/id_rsa.pub`
4. Send the `id_rsa.pub` file to Alex.

Note that you will also have a file named `id_rsa` in the same directory. This is known as your private key and should **not** be shared with anyone.

3.2 Setting Up Git

Git is a tool for keeping track of changes in project files, and for coordinating work between multiple people across the team. You can think of it as a more sophisticated version of Google Drive, or Microsoft OneDrive.

To get started, first check if you have git installed, and download it if it is not installed.

- On **Windows**, open the Start Menu and search for git. Unless you have used git before, it probably will not be installed. Visit <https://git-scm.com/download/win> to download and install git.

To check if it installed correctly, open the Start Menu and search for git. After opening git, a command prompt should open. Now run `git --version`.

- On **Mac**, you may or may not have git installed. You can check by opening a terminal and running `git --version`. If git is not installed, visit <https://git-scm.com/download/mac> to download and install git.

To check if it is installed correctly, open a terminal, and run `git --version`.

- On **Linux**, you may or may not have git installed. You can check by opening a terminal and running `git --version`. If git is not installed, visit <https://git-scm.com/download/linux> and follow the instructions.

To check if it is installed correctly, open a terminal, and run `git --version`.

3.3 Command Prompt Basics

There are numerous commands that can be run in the command prompt. The following are the most useful to get you started.

<code>ls</code>	Lists all the files and directories in your current directory.
<code>pwd</code>	Prints the location of your current directory.
<code>cd directory_name</code>	Navigates to the specified directory (relative to where you currently are).
<code>cd ..</code>	Navigates to the parent directory (relative to where you currently are).

3.4 Cloning the Repo

We first need to download the project files. In the git command line, navigate to somewhere you want to download the project to, and then run the following command.

```
git clone git@directdaylight.com:direct-daylight-game
```

This will create a new folder named `direct-daylight-game` and download the project into that folder.

3.5 Git Primer

3.5.1 Committing Files

Once you have made changes to the project files and are happy with your changes, you are ready to **commit** your changes. Firstly you will need to add the files to the commit. To add all (modified) files to the commit, navigate to the top folder in the project and run the following.

```
git add .
```

Here the `.` indicates that you are adding everything in the current directory.

You can also add individual files or directories.

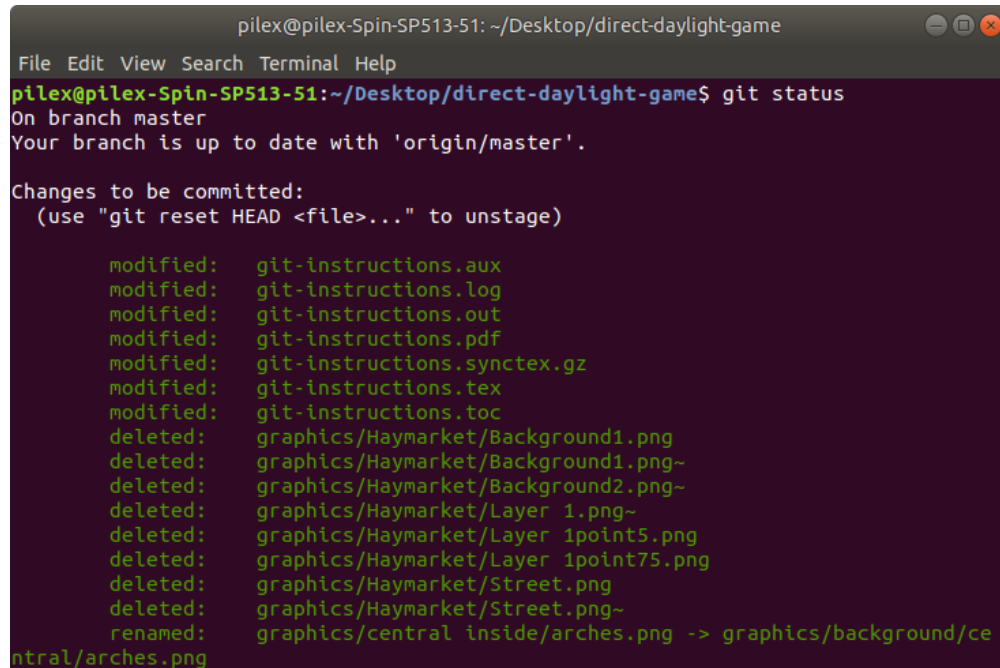
```
git add background2.png
```

To check that you have added the correct files, run the following command.

```
git status
```

This displays basic information about the current state of the project, such as any changes to files you have made, and information about commits.

If you have successfully added your files to the commit, you should see a message like the following.



```
pilex@pilex-Spin-SP513-51: ~/Desktop/direct-daylight-game
File Edit View Search Terminal Help
pilex@pilex-Spin-SP513-51:~/Desktop/direct-daylight-game$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   git-instructions.aux
        modified:   git-instructions.log
        modified:   git-instructions.out
        modified:   git-instructions.pdf
        modified:   git-instructions.synctex.gz
        modified:   git-instructions.tex
        modified:   git-instructions.toc
        deleted:    graphics/Haymarket/Background1.png
        deleted:    graphics/Haymarket/Background1.png~
        deleted:    graphics/Haymarket/Background2.png~
        deleted:    graphics/Haymarket/Layer 1.png~
        deleted:    graphics/Haymarket/Layer 1point5.png
        deleted:    graphics/Haymarket/Layer 1point75.png
        deleted:    graphics/Haymarket/Street.png
        deleted:    graphics/Haymarket/Street.png~
        renamed:     graphics/central inside/arches.png -> graphics/background/central/arches.png
```

Figure 2

If this is your first time using git, you will also need to run the following commands. Replace the email and name with your real email and name.

```
git config --global user.email "johnsmith@gmail.com"
git config --global user.name "John Smith"
```

To commit the files you have added, you will need to provide a commit message, which is a short but descriptive message for describing what you have changed in this commit e.g. "added animation".

```
git commit -m "added animation"
```

After you have committed your changes try running `git status` again, and you should see a message like the following.

```

pilex@pilex-Spin-SP513-51: ~/Desktop/direct-daylight-game
File Edit View Search Terminal Help
pilex@pilex-Spin-SP513-51:~/Desktop/direct-daylight-game$ git commit -m "added a
nimation"
[master 1c9f097] added animation
69 files changed, 1042 insertions(+), 726 deletions(-)
rewrite git-instructions.aux (70%)
rewrite git-instructions.out (92%)
rewrite git-instructions.pdf (68%)
rewrite git-instructions.synctex.gz (98%)
rewrite git-instructions.toc (84%)
delete mode 100644 graphics/Haymarket/Background1.png
delete mode 100644 graphics/Haymarket/Background1.png~
delete mode 100644 graphics/Haymarket/Background2.png~
delete mode 100644 graphics/Haymarket/Layer 1.png~
delete mode 100644 graphics/Haymarket/Layer 1point5.png
delete mode 100644 graphics/Haymarket/Layer 1point75.png
delete mode 100644 graphics/Haymarket/Street.png
delete mode 100644 graphics/Haymarket/Street.png~
rename graphics/{central inside => background/central}/arches.png (100%)
rename graphics/{central inside => background/central}/furniture.png (100%)
rename graphics/{central inside => background/central}/ground.png (100%)
rename graphics/{central inside => background/central}/lights.png (100%)
rename graphics/{central inside/terminalboard.png => background/central/termina
l-board.png} (100%)
create mode 100644 graphics/background/haymarket/background1.png

```

Figure 3

3.5.2 Pushing Your Own Changes

You have now committed your changes to the project, however your changes are not yet visible to the whole team. At this point, you can make more changes and make more commits if you would like. Once you are ready, you can share the changes with the whole team by running the following command.

```
git push origin master
```

This pushes your commits to the main server. Think of it as uploading your changes.

Under certain situations, you may not be able to push to the server. To solve this, read the following sections.

3.5.3 Pulling Other People's Changes

Other people may be working on the project as well, and will be committing and pushing changes using `git commit` and `git push origin master` the same way that you have been doing. Git does **not** automatically download these changes. To retrieve the latest version of the project, run the following command.

```
git pull origin master
```

This may fail if you have edited files but not yet committed them. This is because these changes may potentially be overwritten. The simplest way to solve this is to commit your changes. Alternatively, you can **stash** your changes. We will discuss stashing later on.

If you are having trouble pushing to the server, it may be because someone else has already made changes using `git push origin master`, and you are still on the old version of the project. In this case, running `git pull origin master`, and then pushing your changes as usual using `git push origin master` should fix the issue.

However, if someone else has changed a file that you have worked on, or has deleted a file that you have worked on, `git pull origin master` will fail. In this case you will get a **merge conflict**. We will discuss how to solve merge conflicts in the next section.

3.5.4 Merge Conflicts

A **merge conflict** typically occurs when two people have changed the same lines in a file, or if one person has deleted a file whilst somebody else was modifying it.

More info: <https://www.atlassian.com/git/tutorials/using-branches/merge-conflicts>

3.5.5 Stashing

Stashing temporarily shelves (or *stashes*) changes you've made, so that you can work on something else, and come back and re-apply those changes. This is useful if you are working on something, but you aren't quite ready to commit and want to work on something else.

More info: <https://www.atlassian.com/git/tutorials/saving-changes/git-stash>

3.5.6 Viewing the Project History

You can view the history of the project using the following command.

```
git log
```

This lists out all the commits so far including the commit author and commit message of each commit.

4 Code

The game code is written in Java, using the Processing graphics library. You can find some introductory resources at <http://directdaylight.com/learn-processing/learn-processing-presentation.pdf> to get you started.

4.1 Eclipse

To edit the code and run the game, it is recommended that you use an Integrated Development Environment (IDE) such as Eclipse. The following section will go through setting up Eclipse.

4.1.1 Setting Up Eclipse

1. You will first need to download Eclipse from <https://www.eclipse.org/downloads/> and install it. Inside the installer, make sure you choose the Java edition.
2. When you first run Eclipse, you will get a prompt asking for a working directory. You can pick any folder you want, or just use the default.
3. Once Eclipse has loaded, you will be greeted with the Welcome Page. Click on the *Workbench* button in the top right.
4. Now in the *Package Explorer* tab on the left, right-click and select *Import*.
5. Select *General*, then *Existing Projects Into Workspace* and click *Next*.
6. Select *Root Directory*, then click on *Browse*. Navigate to where you have saved the `direct-daylight-game` folder. Make sure that under the *Projects* tab, `direct-daylight-game` is selected. Also make sure that under the *Options* tab, the *Copy Projects Into Workspace* checkbox is **not** checked.
7. Finally, click *Finish*.

4.1.2 Running the Game

The `main` function is located under `main/Game.java`.

In Eclipse, navigate to `direct-daylight-game/src/main/Game.java`, then click on the green run button located on the top toolbar. This will run the game!