|  |
| --- |
| Purdue University |
| **Destroy the Flags** |
| Game Design Document |

|  |
| --- |
| Corey Abshire, Jory Grillo, Derek Pike  9-6-2013 |



Contents

[Introduction 0](#_Toc367279958)

[Player Personas 0](#_Toc367279959)

[Casual Carrie 0](#_Toc367279960)

[Hardcore Harry 1](#_Toc367279961)

[Art Concept 1](#_Toc367279962)

[Game Structure 2](#_Toc367279963)

[Control 2](#_Toc367279964)

[Scoring 2](#_Toc367279965)

[Game Pieces and Actions 2](#_Toc367279966)

[Boulder 2](#_Toc367279967)

[Flag 2](#_Toc367279968)

[Obelisk 2](#_Toc367279969)

[Reflector 3](#_Toc367279970)

[Slingshot 3](#_Toc367279971)

[Torch 3](#_Toc367279972)

[Additional Rules 3](#_Toc367279973)

[Player Interface 3](#_Toc367279974)

[Screens 3](#_Toc367279975)

[Title Screen 3](#_Toc367279976)

[Player Select Screen 4](#_Toc367279977)

[Main Game Screen 4](#_Toc367279978)

[Screen components 5](#_Toc367279979)

[Artificial Intelligence 5](#_Toc367279980)

[References 5](#_Toc367279981)

# Introduction

Destroy the Flags is a 2 player, turn-based strategy game for mobile phones. The game is set on an 11x9 grid of squares, the top-most two and bottom-most two of which are populated with the player’s pieces. Players take turns of 3 moves each, of which a move can be composed of various units’ moves. The game is distinguished by its unique piece characteristics and capabilities. Due to its similarity to chess, checkers, and other such games, this game will be immediately accessible to casual gamers. Furthermore, due to the size of the game board and rich piece profiles, the high skill ceiling means it will also appeal to hardcore gamers as well.

This game is based directly on a game for Amiga computers from 1990 called Field of Domination, by Mike Duppong. The game won a $10,000 prize awarded during a competition held by a magazine called Amiga Resource. An online archive of an old Compute! Magazine article served as a key guide to recollection of the game design, in addition to one of the team members own personal memory of playing the game back in the day.



# Player Personas

## Casual Carrie

Carrie is a 30-something mother of 2 who throws a game on her phone occasionally to burn a few minutes while waiting at the dentist, or for her kids to finish up soccer practice. She picks things that tend to be more popular and doesn’t have a lot of patience to figure the game out. If it’s not intuitive, she won’t play it and will move on quickly to something else. However, if a game is easy to pick up Carrie is likely to spend considerable time playing it if it turns out to be fun. Carrie’s bought her kids a Wii for Christmas one year and every so often she’ll play Mario or Wii Sports with them. Carrie has a pretty decent modern phone through Verizon, with 4G. It’s a year old now, but still keeps up, and she’s planning on upgrading as soon as it’s available through her contract.



## Hardcore Harry

Harry is an early-20’s gamer that prides himself on his game playing ability. Harry started playing chess when he was 6 and has been pretty good ever since. Harry has a PlayStation 3, an Xbox 360, and a really fast PC with a great graphics card. He gets all the latest games, and writes from time to time to game review websites. He also writes to his personal blog about games he plays. There are a few games in particular at which Harry considers himself an expert. Harry is picky about his video games and doesn’t want to play any lame ones. He can’t tolerate poor game play, but if a game plays well and has a high skill ceiling where he can really challenge himself again players of equal caliber he’s really into it and may dedicate years to mastering it.



# Art Concept

The game art will resemble somewhat closely the original game, but with some enhancements.

|  |  |  |
| --- | --- | --- |
| C:\Users\Corey\Dropbox\Learning\MobileAppDev\TeamProject\ConceptArt\quidditch.jpg | C:\Users\Corey\Dropbox\Learning\MobileAppDev\TeamProject\ConceptArt\Quidditch_Entire_Field_8x6.jpg | C:\Users\Corey\Dropbox\Learning\MobileAppDev\TeamProject\ConceptArt\harrypotter_hbp_quidditch_3.jpg |
| C:\Users\Corey\Dropbox\Learning\MobileAppDev\TeamProject\ConceptArt\IU_FB_PURDUE0126.jpg | C:\Users\Corey\Dropbox\Learning\MobileAppDev\TeamProject\ConceptArt\h.jpg | C:\Users\Corey\Dropbox\Learning\MobileAppDev\TeamProject\ConceptArt\iu_fb_purdue0642.jpg |

Figure . The color scheme of the game will be inspired by Harry Potter, and a Purdue vs. IU football game.

* The game will be brighter and use more vibrant colors than the original. Watching the game should feel like watching a quidditch match on Harry Potter.
* The color scheme will also resemble a Purdue vs. IU football game. Like the original, the game board will look like a light grass / dark grass checkerboard. The piece colors for the “home” player will be black and gold, and the opposing player will be red and white. There will be no option for the user to change this, because we are Purdue, so Purdue is always the good guys in this game.
* The pieces will use pseudo-realistic looking textures and models. This means that for a torch, for instance, it should look somewhat like a wooden torch, but not completely lifelike.
* The spells and effects will be rendered using particle systems and shaders to make the game more engaging.

# Game Structure

## Control

To do.

## Scoring

There is no score in the game. A player wins by destroying the flags of the opposing player. An opposing players flags can be destroyed by crushing it with a boulder, burning it with a torch, shooting it with a rock, or various destructive spells from the obelisk. When both of a players flags have been destroyed, that player loses and the other player wins.

# Game Pieces

## Boulder



* Takes 4 slingshot hits to destroy.
* Cannot move onto another boulder, unless that boulder has already been reduced to rubble.
* Unaffected by fire and water spells.

## Flag



* The main type of piece to be protected.
* Takes only one hit with a slingshot to destroy.
* Cannot move onto the same square as any other intact piece.

## Obelisk



* Takes two hits with a slingshot to destroy.
* Can shoot spells.
* Each spell cast takes magic points to cast.
* Each player has a given amount of magic points that regenerates by some degree per turn.
* Can be destroyed by boulders.
* Cannot move onto the same square as another intact piece.

## Reflector



* If hit at other than a right angle, reflects the shot.
* If hit at a right angle, is destroyed.

## Slingshot



* Each slingshot can fire one rock per turn.
* Slingshot rocks can hit all other pieces.
* Slingshot rocks are reflected by reflectors.
* Can be burned down by the fire spell.

## Torch



* Starts out in a lit form.
* If lit, can move onto a square containing a wooden piece (slingshot, flag, or block) to burn it.
* Can be extinguished by the water spell.
* Can be relit by the fire spell.

# Spells and Abilities

Two of the pieces have additional capabilities beyond the basic movement mechanic: the obelisk and the slingshot. The obelisk is like a wizard, with magical abilities in the form of powerful spells it can cast towards enemies. The slingshot has the ability to shoot rocks at enemy pieces. Both abilities fire in the form of projectiles that travel across the screen. These pieces are also able to rotate in 45 degree units giving it the ability to fire in 8 different directions.

## Fire



* Lights torches that are no longer lit.
* Can burn slingshots and flags to ashes.
* Has no effect on boulders.
* Burns reflectors to ashes if it hits them from the side.
* Costs 2 energy.

## Water



* Puts out fire on torches.
* Has no effect on other pieces.
* Costs 1 energy.

## Roots



* Anchors the target to the ground until the effect is removed with a rejuvenate spell from the obelisk. During this time, the piece cannot move.
* Does not work on boulders.
* Costs 3 energy.

## Shield



* Protects its target for some period.
* If used on a torch, the flame is extinguished.
* While protected by a shield, the piece cannot move.
* A shield deflects both enemy and friendly spells.
* A shield does not prevent pieces that are capable of crushing from doing so.
* Costs 3 energy.

## Stun



* Causes a piece to be unable to move for a few turns. <need to define>
* Has no effect on boulders.
* Costs 3 energy.

## Rejuvenate



* Heals a piece that has been damaged by slingshot rocks.
* Heals only one stage of destruction at a time. That is, for a boulder which takes 4 shots to destroy with a slingshot, the rejuvenate spell would also take 4 shots to heal it completely.
* Costs 4 energy.

## Rock



* Fires a rock from a slingshot.

# Additional Rules

* If a piece has been destroyed, it leaves rubble on the map. This can be restored to the original piece by casting the rejuvenate spell on the rubble from the obelisk.
* If another piece moves over the rubble of another piece, the rubble is removed from play.

# Player Interface

The game is configured for modern (~ year 2013) smartphones, most of which seem to have a 16x9 aspect ratio. We divide the game screen into a 16x9 grid. The 11x9 game grid is put

# Screens

The game UI is comprised of a handful of screens that the user navigates among. Screens all occur within the same activity. Whereas a normal Android app might break down a design involving multiple such screens as simply multiple activities, we need tighter control over the transitions and have to bind with our own animation updates anyway so we model it with our own screen system within an activity. The figure below shows the basic overall flow the user can take from screen to screen.



In the next sections, we describe each screen.

## Title Screen

The title screen is the main screen presented to the user when they aren’t playing a game. After any splash screens have cycled through when the game is initially launched, the app comes to the title screen to await user input.

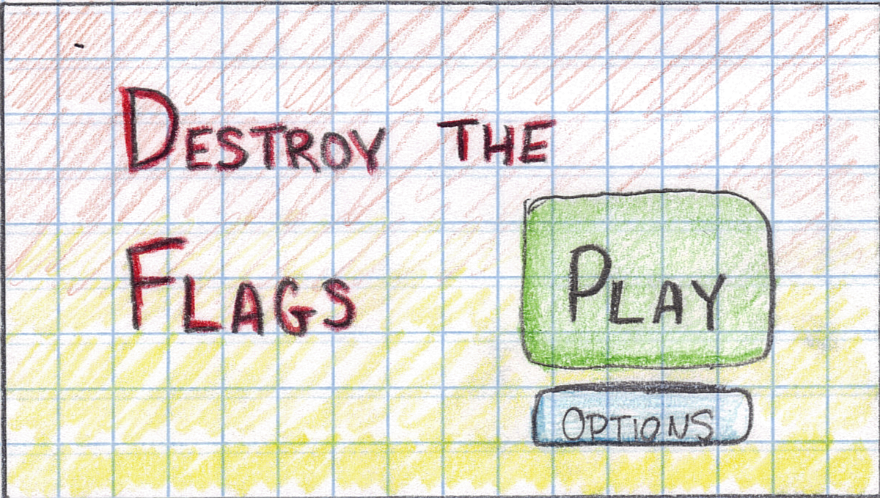


Figure : Title Screen.

## Player Select Screen

The player select screen is shown when the player clicks play from the title screen. It allows the player to choose between one player mode vs. the computer and two player mode with both players sharing the same device as if it were a real board game.

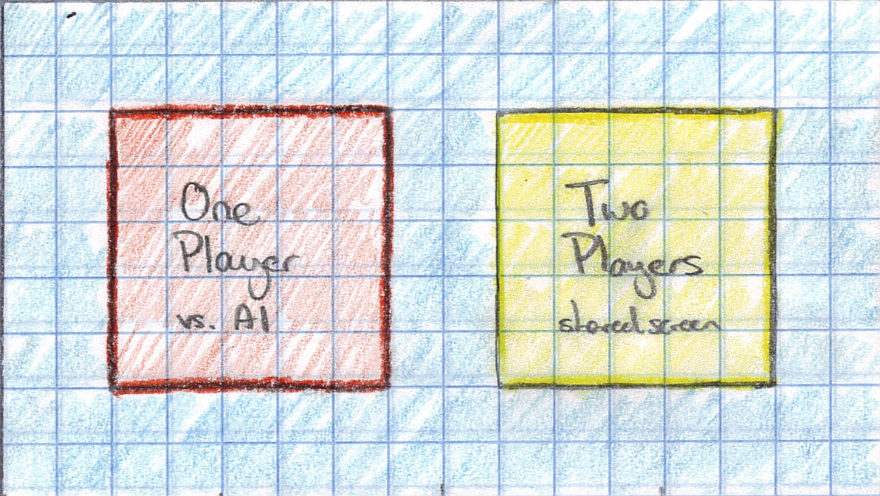


Figure Player Select screen. Players use this screen to choose between one player mode and two player mode.

## Main Game Screen

The main game screen shows up after the player chooses whether they are going one player or two player. This is the main screen at which the player plays the game. The screen looks the same, regardless of whether it is one player or two player. The only difference is that in one player, the screen does not accept the move from the touch interface, and instead the computer makes the move.

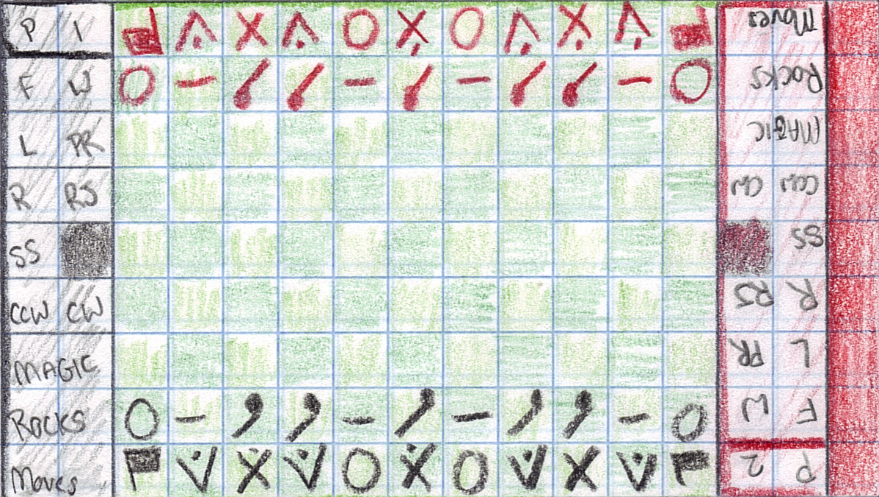


Figure . Main game screen. This is the main screen where the game is actually played.

### Screen components

* The main board area is an 11x9 green grid in the middle of the screen. This is where the pieces are located and where the two players compete.
* There is one sideline for each player to either side of the main board area. This area contains
  + the buttons for the spells that can be cast from the obelisk
  + the button for shooting rocks from the slingshot
  + buttons for rotating clockwise and counter-clockwise for rotatable pieces.
  + the amount of magic remaining for casting spells
  + the number of rocks remaining for firing from the slingshots
  + the number of moves remaining for this player this turn
* To the far right of the screen from the black players perspective is the turn indicator bar. This bar turns black when it’s the black players turn and red when it’s red’s turn. In the figure above, it is currently red’s turn.

## Additional Notes

* There will be one main omnidirectional light source centered just above the game board to mimic the sun.
* The following objects will also serve as light sources:
  + Lit torches
  + Spells in flight (color depends on spell)
  + Magic “orange” (protective rings and reflective parts of blocks)

# Assets

The following table describes all the assets we will need for the game.

## 3d Models

* Torch
* Boulder
* Rock
* Obelisk (a.k.a. wizard)
* Square
* Slingshot
* Reflector
* Flag
* Selection

## 3d Shaders

Each shader consists of an OpenGL ES 2.0 fragment / pixel shader pair.

* Normal game shader
* Fire shader
* Water shader
* Lightning shader
* Protective ring shader
* Roots shader
* Rejuvenate shader
* Rock shader
* Flag shader
* Slingshot band shader

## Textures

* Light grass texture
* Dark grass texture
* Gold piece texture
* Red piece texture

## Additional Notes

### Preferred 3D file format

There are many 3D file formats available and all have their pros and cons. JPCT provides good support for about 3 or so of the most common ones. The two main alternatives that are appealing for our workflow are 3DS and Wavefront OBJ. The latter is usually used for a lot of work like this because it is a text readable format and can be read by about everything. Unfortunately because it’s a text based format the files can be fairly large, and the way its structured it needs multiple files to describe a game object (the main object geometry file and one or more materials files). 3DS is a better choice here because it is a more efficient binary format and holds all the geometry and materials in the same file. This will make it a little more efficient for us and still be mostly convenient since it’s so common.

### Preferred texture file format

PNG’s would work best for our textures because we can avoid compression effects, still be reasonably efficient and use alpha channels. The built-in support for it is good both within Android libraries and JPCT.

# Supported Devices and O.S. Version

The game will try to support all Android devices with at least version 7 (Android 2.1) of the Android SDK.

<need more thought here>

# Game Board Text File Format

A useful utility for game development and testing for this type of game is a simple text format that looks like an text version of the game state. For it to be really useful, we should agree on a thorough but convenient format. This will allow us to build a standard set of readers and writers based on the format, and a repository of various game scenarios, including the initial board configuration. Once this is in place, we can take advantage of it to build various test utilities and other to be defined capabilities to help us throughout the development process. Below is a sample text file based on a format proposed for this purpose for this game.

## Sample File

RF04 RV04 RX04 RV04 RB04 RX04 RB04 RV04 RX04 RV04 RF04

RB04 RR04 RT04 RT04 RR04 RT04 RR04 RT04 RT04 RR04 RB04

\*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\*

\*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\*

\*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\*

\*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\*

\*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\* \*\*\*\*

GB00 GR00 GT00 GT00 GR00 GT00 GR00 GT00 GT00 GR00 GB00

GF00 GV00 GX00 GV00 GB00 GX00 GB00 GV00 GX00 GV00 GF00

## File Description

The file looks like the game in action. There are 11 x 9 cells, with 9 lines of text each having 11 strings describing the square state. Each string is composed of 4 components:

1. R or G: describes whether the piece on this square is owned by the red or gold player.
2. F, V, X, B, T, R: describes the type of piece on this square
   1. F: a flag
   2. V: a slingshot (V looks similar to the slingshot graphic)
   3. X: obelisk (X looks similar to the obelisk graphic)
   4. B: boulder (perhaps O would have been more appropriate?)
   5. T: a torch
   6. R: a reflector
3. 0 .. 4: The number of hits of damage this piece has taken.
4. 0 .. 8: The direction this piece is facing.  
   

The \*\*\*\* fields are simply empty squares.

One weakness of this format is that it’s currently unable to represent extended piece state such as spell effects, whether torches are lit or not and so forth. Initial game state should be assumed the default (e.g., torches are to be assumed to be lit). The tradeoff here is convenience. We could add fields to describe extended state but it can get a bit unwieldy and perhaps doesn’t add as much value? This is something we should discuss sometime soon.

# Coding and Team Standards

* Since we’re all using Eclipse for the class already, let’s favor using the default formatting conventions it uses when you select the code and use the built-in formatting.
* Let’s turn on the 80 character margin indicator and try to roughly stick to it. The Eclipse formatting rules also do this, though we can skip it where it makes sense.
* Let’s omit author tags in Javadocs throughout the repository. We’ll know who wrote what through git, and the rest of the credits can be in the game documentation and so forth.
* Let’s use git and github for our version control, along with the egit module for integration with Eclipse.
* Let’s follow an agile development style of working together, we’ll use Trello for tracking our backlog for agile development.
* Let’s be sure to put appropriate Javadoc comments on all public interfaces, classes and methods, and regular comments where they make sense throughout the rest of the code.
* Try to be sure we don’t pull into the main code branch anything that breaks the build or has bugs or messy code in it.

# Artificial Intelligence

To do.

# References

Duppong, M. (1990). Field of Domination. Compute! Magazine. Issue 123. Page A-27. Retrieved from: <http://www.atarimagazines.com/compute/issue123/FIELD_OF_DOMINATION.php>