Abstract:

For the past decade, the multiplayer online battle arena (MOBA) has been one of the most popular genres of video game. Titles like Defense of the Ancients 2, League of Legends and Smite have millions of active players that enjoy the game on a daily basis. MOBAs provide hours of entertainment through fun, flashy characters, thrilling decision making, and satisfaction from mastery of characters. While some attempts have been made in the past to bring this genre to mobile with varying degrees of success, they often try to recreate the full intensive gameplay of the PC instead of playing to the strengths of mobile devices. Oathkeepers aims to bridge the gap between traditional MOBAs and the mobile space by keeping the core elements of the genre and streamlining the rest. I created Oathkeepers as a proof of concept game for the Android that demonstrates the approach to mobile design by using an intuitive control scheme of tapping to select the board, keeping game length to 10-15 minutes instead of the usual 40+, and having one player control an entire team instead of 4 players controlling one hero each. At the same time, it delivers on the MOBA promise of interesting heroes, fun combinations, and strategic decision making. The Team Builder allows players to customize their team of heroes, which changes strategy from game to game. Simultaneous turns emphasize coordination among your team and predicting your opponent, while still allowing for surprise and a more relaxed control scheme. Multiple playtests throughout the development process provided user feedback that helped improve the experience.

Game Description:

Oathkeepers pits two teams of four heroes against each other in a battle to determine which team can either destroy all of the enemy heroes or the enemy base. Having an immobile target like the base mirrors the MOBA’s goal of pushing down lanes to get to the enemy team’s base. Similarly, the smaller team size of 4 places a greater emphasis on the choice of hero. Players begin by going to the Team Builder and drafting their teams of 4. Players can have any combination of heroes, which are saved upon pressing the Ready button.
Once the teams are assembled, players launch the main game by pressing the Play button. The game begins with the heroes placed on either side of the board. From there, both players begin issuing orders to their heroes, which is controlled by a series of taps. Tapping on a hero brings up their ability menu, and tapping one of the abilities shows the legal selections on the game board. Tapping one of those selects that square as the target. When a player has given their commands, they “Ready” their base. All selected actions are processed simultaneously when both bases are ready.

The turns repeat until one player has achieved victory. Then, the players have the option of either an immediate rematch, or returning to the main menu, drafting a new team of heroes, and jumping back into the fray!

**Design Process:**
At first, I was working directly with Android’s Activity class. This is the basic class that represents the most fundamental way of interacting with the Android, and as such, was mainly geared towards more typical apps that were designed to display xml classes. Therefore, I had to make my own Game class, graphics renderer, input processor, and so on. In other words, before I could make a game, I had to make a game engine. This process was slow, taking up time that could have been spent on game-specific logic, and wasn’t terribly efficient.

These early stages were focused around rendering a board, assigning heroes to that board, processing inputs, and moving heroes.

Fortunately, I managed to find a game engine specifically for Android development a couple of weeks into the project. While that meant that the engine work that I had done ended up being a waste of time, LibGDX was a powerful tool that overall led to a much more productive development process. I ported over what game-specific logic I had and started over in the new framework.

LibGDX Framework:

LibGDX provides features that are common to most games. The Game class acts as an Activity, but delegates responsibility to Screens, another provided class, that further interact with Worlds, Renderers, and InputHandlers, which are classes that I write, but make use of LibGDX methods.

To use the main GameScreen as an example, when the player selects the Play button on the main menu, OathGame, the overall game class, sets the to GameScreen, which instantiates GameWorld, GameRenderer, and InputHandler classes. For each frame, the GameScreen calls GameWorld’s update function, which contains each of the game objects. This would be the heroes, the board, and the ability bar. During the turn itself, whenever a player taps the board to issue an order, the InputHandler is called, where I have written logic to determine what square is selected and how to resolve that action.

Once all of the objects have been updated, GameScreen calls GameRenderer’s render function, which handles drawing the shapes, sprites, and animations onto the screen. This art, in turn, is brought into the game with the static AssetLoader class. Both of these classes are written by me, but make use of
LibGDX graphics functions that turn textures into sprites and animations, selects keyframes, and draws pixels.

So, LibGDX provides low-level support for features common to all games, such as input detection, pixel drawing, communication between game and screens. However, all game-specific code such as the heroes and their abilities, the boards, how inputs are processed, all art, and how each screen is designed was provided by me.

**Gameplay Elements:**

*Team Composition* – Deciding which heroes to put on the team and in what position is the first part of gameplay. Each brings different strengths and weaknesses to the field. For example, the Knight (Oath of Justice) has high armor and damage, but is a melee character. The Abomination (Oath of Plague) is more fragile, but has greater range to stay out of damage. In addition, since the heroes are placed on the board in order, it makes a difference in strategy. Are tough heroes placed in the center to defend the base, or ranged heroes to quickly threaten the opposing side?

*Strategic Decisions* – Because each hero can only perform one action per turn, the choice between attacking and moving becomes more important. Attacks have priority over movement, so a player can guarantee damage against an enemy, at the cost of staying in place. Also, does the player want to focus on eliminating one enemy in a turn by focusing attacks on them, spread damage across the team, or go directly for the base? Depending on the hero’s preferred strategy, this may impact the heroes drafted in the team builder.

*Positioning* – In games of this genre, this is the concept of “developing the board.” Is it more important to move into range of the enemy base, move to the defense of an ally, or attack an enemy? With the Knight, he must move into range, but can then attack any square around him. This makes him good at defending an area. The Abomination has longer range, but only in certain directions, requiring the player to predict the enemy’s movements.

**Future Goals:**

*Expanded Roster* – The Knight and the Abomination are two of the many planned heroes that will increase the depth of strategy and team building by bringing new abilities to the roster.

*Expanded Abilities* – While the interplay between melee and ranged leads to interesting gameplay, more complex abilities will enable deeper and more unique strategies.

*Play Beyond Local* – In the current state, players can compete and test their strategies and decision-making against their friends, but opening up play online would allow more intense competition.