Due: 30 Oct 2013 **CS 344 Final Project Proposal** Andrew Shim

Update Since Initial Proposal

This document is pretty much the same as the Initial Project Proposal, but I have a couple updates. First, I have decided to use Vuforia, which is an AR SDK. Given this, I will likely not use OpenCV, because Vuforia provides image recognition out of the box (although I will have to tweak it to my needs). Also, I have set up an XCode project and a GitHub repository:

`https://github.com/andrewkshim/Assassins`

The rest of this document is the same. I am slightly behind my outlined schedule but I set in some buffer time so I'm still projected to finish on time.

Description of Final Product

I want to make an augmented reality (AR) game for the iPhone. Here are some samples of AR games for mobile devices:

- AR Defender
- Zombies Everywhere!

My game is going to be an AR adaptation of the game Assassins and will go by the same name for now (I know its no "Peckerman" or "Sil y Vega Peck"). The basic premise of `Assassins` is similar to Tag but with a couple of "killer" modifications; you gather a group of players then the game host/referee will assign each player a target (another player) to "assassinate". Assassination can entail a variety of actions, some examples I have seen are: poking the person in the back with a spoon (basically simulating that you're stabbing them, I know it is violent but the game's intentions are completely harmless I swear), touching them with a (clean) sock, and shooting them with a Nerf Gun. Over time, several embellishments to the game have been added. For example, a common practice is to allow players to carry a safety item (usually something cumbersome or embarrassing such as a pair of underwear) that gives them immunity to attacks as long as the item is visible.

My current vision for the game is that users will use their camera viewfinders in order to claim their targets (more details below). The attack will not be instantaneous, instead the target will receive a notification about the attack and will have the opportunity to evade it if they notice in time. The attackers' cameras will be able to recognize their targets because every target will be required to wear a brightly colored square (they can print these out themselves) that must be clearly visible. If the square is not visible, the attackers can attack by taking a photo of their targets and sending it to the game host, who can then confirm the attack and eliminate the target. The safety item will also be a part of the game and will be implemented with brightly colored squares as well (players will need to get used to wearing squares). If the attacker's camera sees the safety item on the target, the attacker will not be able to attack.

This is the very basic functionality I want in the game. Time permitting, I plan on adding features such as traps and powerups. An example of a trap is a landmine, which a player might place in a highly trafficked area to blow up unsuspecting players. The landmines would use GPS coordinates, so if a player was within some trigger distance to a landmine they would get eliminated. A powerup example is a one-time use shield, allowing a player to survive an attempt on their virtual life.

Using Computer Graphics

One cannot have AR without computer graphics (CG). `Assassins` will use various computer graphics to enhance the player experience.

CG in the Attacker's View

As attackers are viewing their targets through their phone cameras, if the target is wearing the safety item (the colored square) I will render an aura of light around the target, which will be done through a particle generator. Also, I will to use computer vision (OpenCV) to detect the targets and form a bounding box around them. Once attackers have their target on their screens, they will need to tap on the target and "shoot" them down. A tap will send out an energy ball that will get smaller as it goes farther into the negative z direction. Once the sphere is small enough and if it is still within the target's bounding box, the target will have been hit.

CG in Any Player's View

If I get to implement powerups, I will make them such that powerups appear at random GPS coordinates. Note that I plan on having the referee bound the game map to a square defined by four GPS coordinates so if a group of Duke students are playing, a powerup will not pop up in Venice. Players will be notified if they are near a powerup, after which they will need to pull out their phone cameras. If they are in the correct location, their camera will show a visible sparkle (more particle generators) in the center of the screen and they will pick up the powerup, after which the powerup will no longer be available to other players.

Related CG Topics

The CG enhancements fall under a range of topics we have covered in class. Using the energy balls as the primary example, I will need to set up the **material properties** to make them look like this (top photo). If you are reading this as a print out then you probably cannot click the link, so just imagine the energy balls as shiny blue balls of light. Also, as the energy balls travel in the scene, I will make it such that they emit light (**emissive**), and to make it realistic I will modify the scene's **shading and lighting** based on the energy ball's location. To make them look even cooler, I will use a **particle generator** to give them an aura. I plan on using a combination of **OpenGL** and iOS 7's new SpriteKit framework to implement all of these effects.

Who is Working on the Project

Me! I am flying solo on this one. I will do all the things!

Hardware Requirements

I have an iPhone to test the app and a Mac to develop on; I will not need any additional hardware.

Project Timeline

- October 25 - Create XCode project, begin core backend
  - Set up User and Round models
    - User can be a Host or a Player

        * Allow Hosts to create a Round
        * Allow Hosts to invite Users to a Round
        * Allow Hosts to assign targets to Users

- October 28 - Begin writing final project proposal
- October 30 - Final project proposal due, download OpenCV for iOS
- November 1 - Finish core backend, begin UI, weekly progress check
    - Make sure everything from October 25th is working and bug-free
    - Allow Players to recognize their targets
    - Use OpenCV to recognize brightly colored squares in the scene
        * Implement logic to prevent Players from attacking when the target's square is visible
        * Place an aura particle generator at the location of the square
- November 8 - More UI, start backend for cooler features, weekly progress check
    - Allow Players to fire energy balls at their targets
    - Use OpenCV to create bounding bounding boxes around the target
    - Implement logic to determine a successful attack
    - Remove Player from Round if they have been hit
- November 15 - Catch up (buffer time in case there is a delay in the schedule), weekly progress check
- November 22 - Begin final project write up, polish code and iron out all current bugs, weekly progress check
- November 29 - Finish final project write up, begin bonus features (e.g. traps and powerups), weekly progress check
- December 6 - Final project write up due, finish bonus features, weekly progress check
- December 13 - Practice demo, iron out all remaining bugs
- December 14 - Demo