# OnTheRun: A Location-based Exercise Game

by

Matthew Donahoe

Submitted to the Program in Media Arts and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2011

Author . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Program in Media Arts and Sciences
August 5, 2011

Certified by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Christopher Schmandt
Principal Research Scientist
MIT Media Lab
Thesis Supervisor

Accepted by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
Mitchel Resnick
LEGO Papert Professor in Learning Research
Academic Head, Program in Media Arts and Sciences

# OnTheRun: A Location-based Exercise Game

by

Matthew Donahoe

Submitted to the Program in Media Arts and Sciences
on August 5, 2011, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

## Abstract

Going for a run is a great way to get exercise and feel rejuvenated, but it can also get repetitive and boring. By contrast, digital games can be very engaging and addictive but traditionally force players to be physically inactive. This project is a mobile phone application that transforms a run into an immersive game by presenting a series of story-based running tasks to motivate the player. Requiring that the game be played while running constrains the design in terms of both input and output, and this system demonstrates ways to manage those constraints while still providing a compelling experience. A six person evaluation validates the concept and offers useful design feedback.

Thesis Supervisor: Christopher Schmandt
Title: Principal Research Scientist
MIT Media Lab

Thesis reader . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Dr. Joseph Paradiso

Associate Professor in Media Art and Sciences

MIT Program in Media Arts and Sciences

Thesis reader . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Henry Holtzman

Research Scientist

Chief Knowledge Officer

MIT Media Lab

# *Acknowledgments*

I would like to thank Chris Schmandt for his guidance over the past two years and for giving me the opportunity to be here at the lab. Thanks to my fellow group mates, Jaewoo Chung, Andrea Colaco, Charlie Detar, Drew Harry and Wu-Hsi Li, for their advice. Drew was especially invaluable as a sounding board, writing coach, and cameraman. Charlie O'Keefe lent his voice and creativity to the character Ulysses. Eugene Kozlenko helped create the first-person-view shots for the demo video. Patsy Bailin contributed her running insight and copyediting skills. Tara Krishnan helped develop early tools for visualizing run log data. My father, Steve Donahoe, loaned his iPhone to use as a testing device, wrote an early version of the storyline, and proofread versions of this document. Special thanks to my six testers for their eagerness and patience. Lastly I would like to thank my wife Jessica for her encouragement.

I would also like to thank the OpenStreetMap project and their contributors for making their map data free and easily available.

# *Contents*

# 1 Introduction

*OnTheRun* is a smartphone game that is played outdoors while running. It is intended as a fun alternative to running with music. The game uses an adaptive route planning and navigation system to guide the player along nearby streets, putting the player at the center of a fugitive storyline with narration, sounds and music, for motivation. Running logs are viewable on the web so that players can relive their experiences.

## 1.1 Scenarios

Jack has been out of college for a few years and has settled into a regrettably inactive lifestyle. His job requires him to sit at a computer all day, and many of his leisure time hobbies, such as watching TV and playing videogames, are similarly sedentary. He wants to do more physical activities but has difficulty with solo exercise because he isn't sure what to do. He has gone running a few times but gives up early since no one is pushing him. Teams sports are fun for him, but he does not want to commit to a weekly schedule. Looking to try a new approach, Jack downloads an interesting new application to his smartphone called *OnTheRun*. It is a game that promises to make running more fun by turning each run into an exciting mission.

Jack starts using the app and goes for a run every few days. Each mission has a unique plot that builds on its predecessor. The missions have taken him on many different routes around his house, and he has seen much more of his neighborhood. Jack especially likes that the app provides feedback while he runs, because it pushes him to go further and faster than he would otherwise go on his own. Jack also uses the log viewer to remember past missions and is impressed with the number of runs he has completed. Jack's experience playing the game has made him more confident in his running abilities, and he feels good knowing that it is more active than his usual videogame.

Unlike Jack, Jill is already an avid runner. She runs several miles a day, six days a week if possible. She typically runs the same handful of routes every week and has carefully designed each route for a specific distance and difficulty. To her, running is a great stress reliever and a source of accomplishment, especially when she sets a new personal record. However, there are some days when she wants a break from her routine, but not necessarily from running.

Though she doesn't play videogames regularly, Jill downloads *OnTheRun* and begins to use it occasionally when she wants her exercise to be more exciting. She likes that she can specify exactly how far the app will take her and that it will generate a different route each time. After each run, she looks online to see a log of where she went and has discovered new places to include in her normal running routines.

## 1.2 Goals for this thesis

The goal of this thesis is to create a fun, single-player game that is played while running. There are many active games that involve multiple players, like most team sports, but single player games are mostly relegated to computer screens. Creating an active game that is engaging for a single person and playable while running will be a challenge.

# 2 Background

As work in the developed world has shifted from predominantly physical labor to more information oriented jobs, our daily lives have become far more sedentary. Because we don't get exercise in our normal lifestyle, it is especially important to seek it out. In this work, I have focused on one of the simplest exercises available: running. It is a solo activity, requires no additional equipment, and doesn't rely on the memorization of particular moves. But even running can be complicated for novice runners. How far should they run? How fast? How often? This uncertainty makes even the most straightforward exercise technique problematic for new and unmotivated runners.

One of the major challenges of exercise is managing the tension between a short term discomfort and a long term reward. Successful runners have to develop motivation to bridge this gap. There is a performance difference between intrinsic and extrinsic motivation, and Self Determination Theory, as described by Deci and Ryan (1985), has been applied frequently to exercise and other motivational contexts. The theory suggests that intrinsic motivation ("I like doing this") will outperform extrinsic motivation ("I am doing this for the reward"). So if a runner's motivation for exercise is just to lose weight and be healthier, he is less likely to be successful. This effect is compounded because the rewards of exercise are not quickly evident; a single run does not result in instant weight loss and fitness. A successful runner enjoys running intrinsically and will typically run a very regular routine of many miles per week. However, since running is so simple and repetitive, even an avid runner appreciates variety in his routine and can lack motivation on occasion.

Exercise isn't fun like games. Koster (2005) claims that fun is the feeling we get when we are learning, and games are fun because they present learning challenges in a way that is easily consumed. Games are a form of mental exercise, and our brains constantly work to understand the patterns that drive the game. Games also create a safe environment for players to experiment without long-term consequences for failure. Over time, players learn from mistakes and develop the skills and understanding necessary to accomplish goals and win.

Games have the power to help people accomplish amazing feats by providing the proper motivation, guidance, and feedback. This is why games are a perfect match for exercise. In a summary paper, Whitehead et al. (1993) suggest that low effectiveness exercisers will be more motivated and have better adherence if they feel more competent. Going out for a run and not knowing how far or what to do prevents low effectiveness exercisers from appreciating their accomplishments. If instead they are given an explicit, tangible, achievable goal by a authoritative source, they will understand exactly what they need to do and feel great when they accomplish it. Games can provide that structure.

Sports are the traditional blend between physical and mental exercise. Team sports are fun, healthy, social activities involving long and short-term goals and the acquisition of skill. Unfortunately, since team sports require all members to be physically co-located at a specific time, scheduling conflicts prevent potential players from joining teams. Team sports also tend to require the acquisition of sport-specific skills that make it harder for more expert players to play with less expert players. Solo sports like running solve these problems at the expense of losing most of the game-like qualities that create fun. Instead, many runners add game-like mechanics into their routines by tracking personal records and setting goals.

Games exist in many different forms, but computer[1] games offer a single-player experience unlike any other. Prior to the invention of computers, most interactive games required multiple people in order to create excitement and entertainment.[2] Games are governed by rules, and it is the players who agree on and hold each other to a common set of rules (Caillois and Barash, 2001). Players also get the challenge of competing against each other, but by using a computer, the game itself can hold the player accountable and create challenges. The player can interact solely with the computer, and this allows the player to play the game at any time. It is also less embarrassing to fail in front of a computer than a peer, coach, or instructor, which is why a single player game is a good way to build the confidence of a novice exerciser.

[1] I use "computer" to mean all games that require computation, not exclusively games that are played on desktops and laptops.

[2] Solitaire is probably the most notable exception, though ironically it is mostly played via computer now.

*OnTheRun* is a running game. It challenges the player mentally and physically in a way that is safe[3] and easy to follow. Novice runners will enjoy the direction and feedback it provides, and avid runners will enjoy the new mental challenge it brings to their well-understood routine.

[3] Safety not guaranteed. Beware of cars!

# *3*
# *Related Work*

*OnTheRun* is a location-based, audio-only, alternate-reality, everyday running game that provides turn-by-turn directions. Since it exists at the intersection of several categories, it is related to many different projects.

## 3.1 Exergaming

Exercise games (exergaming) is a rapidly growing field, and there are several mainstream products available on the market, most notably *Dance Dance Revolution*(Konami, 1999), *Wii Fit*(Nintendo, 2007) and recent games for the *Kinect* (Microsoft, 2011). Players use their entire bodies (instead of just their thumbs) to complete active game challenges. *Wii Fit* is explicitly about exercising, and a player's progress and accomplishments can be tracked over several weeks or months. Other games mimic real world activities, such as dancing or fighting, and will quickly get players sweating (even if they didn't intend it). Most exercise games are played indoors on a computer or TV; mobile exercise games like *OnTheRun* are not as common. Getting a computer game to work reliably outdoors is difficult, but running outside is more exciting and worth the effort.

## 3.2 Location-Based Games

Location-based games distinguish themselves from traditional computer games by leveraging the player's real-world location as an additional game input. Thus these games are played on portable or mobile devices with position sensing abilities. The typical embodiment of this device is a GPS equipped smartphone, which is rapidly becoming the dominant computer platform. Games use location input differently, and over the past decade many different games have sprung up in the literature and the marketplace.

*Parallel Kingdoms (PerBlue, 2011)* is a game that uses the player's current location to form a bridge between the real world and a parallel game layer. The player creates landmarks and interacts with other players and their creations in a persistent game world viewable only through the device's touch display. The player can move his avatar in the game world by tapping on the map, but the avatar is constrained by a 100m radius circle tied to the player's current physical location. Thus, if the player wants to move a long distance in the game world, he will have to physically move closer to that location.

Since most of the interaction is screen-based, it is not an effective form of exercise, nor is it trying to be. Players don't run around while playing. Instead they mostly take the game out whenever they have physically moved someplace new during the course of their day. The purpose of *OnTheRun* is exercise, so running is the dominant form of interaction. *OnTheRun* also does not have a persistent world because I didn't want starting location to be a barrier to exercise.

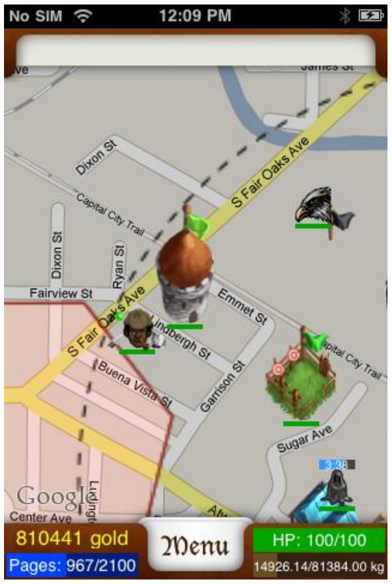


Figure 3.1: A map from *Parallel Kingdoms*.

*CanYouSeeMeNow (Flintham et al., 2003)* is a game in which online players try to avoid capture by runners in the real world. Runners try to surround and corner the players, but the players can listen in to the runners' walkie-talkie channel and move accordingly. This is a scheduled multiplayer event and therefore would be a poor fit for use in a daily personal exercise routine.

*Bystander (Flintham et al., 2003)* has the player track a mysterious person through the city with the help of an online performer. The online performer sees a "zoomed-in" 3D map of the world (tied to the player's current location) and uses this information to guide the player to the next clue via text messages. Using an online performer makes the navigation directions more interesting because the runner knows the messages are being generated by a live human. However this requires a second person and might not work well for daily exercise.[1] *OnTheRun* makes use of a scripted storyline, prerecorded dialog, and navigation algorithms to achieve a similar experience.

[1] An online, real running coach that talks to a runner while they run could be a useful service.

*PiNiZoRo (Stanley et al., 2010)* is a waypoint following game designed to encourage kids to go on walks with their parents. The parents produce a waypoint map that their child follows using a phone, and along the path, the child will encounter enemies that he must fight using an onscreen interface. The concept of non-player generated travel routes is useful, and using parents to generate the routes ensures that the child will be safe. *OnTheRun* attempts to automate the route building process.

*Wanderer (Hielscher and Heitlager, 2006)* is an audio+GPS game that tells the player actions to perform while jogging. The player must maintain a certain speed and perform the spoken actions ("Go left!", "turn around!", etc) within a time limit in order to receive points. *OnTheRun* builds on this experience by incorporating running commands into a larger storyline. The player runs to accomplish mission goals and drive the plot forward rather than to acquire points.

## 3.3 Everyday Fitness Games

Everyday fitness games encourage players to exercise regularly by rewarding daily effort. A game will record players' exercise statistics and convert exercise progress into more visible game progress. These games all attempt to keep the player engaged across many exercise outings, but they make little effort to improve the fun of the activity itself. *OnTheRun* attempts to recreate some of the offline interaction that these apps provide, while also making the running experience very engaging.

*NEAT-o-Race (Fujiki et al., 2008)* is a PDA application that allows players to compete in a virtual race powered by their daily activity. Players can see their activity level affect the game in real-time if desired, but it also works in the background, allowing the player to compete without focusing on the game.

*Fish'n'Steps (Lin et al., 2006)* is a competitive game in which players wear pedometers and use their daily accumulated step count to feed virtual fish. This is a multi-day activity that makes players want to walk each day so they can see their individual fish grow and compete to have the biggest fish.

*Kukini (Campbell et al., 2008)* is a running game based in a virtual world that players explore by undertaking various quests. Players play the game on a computer, but they complete quests by accepting small running challenges, which are measured using the *Nike+* pedometer.

## 3.4 Running Accessories

The rise of smartphones has inspired developers to create numerous running applications. Most of these applications are geared toward existing runners who want to track their runs and capture as much real data as possible for reflection or training. While not explicitly a game, the collection and sharing of data can provide some of the same motivation that I seek to inspire in my work.

*RunKeeper (Runkeeper, 2010)* is an app that includes GPS tracking to record the runner's route and share it online with friends. It can also stream live results to the internet so friends can watch a runner's progress, which is popular for running events like marathons. Run route tracking applications like this one inspired *OnTheRun*'s mission log viewer, which displays the route information alongside a transcript of the mission.

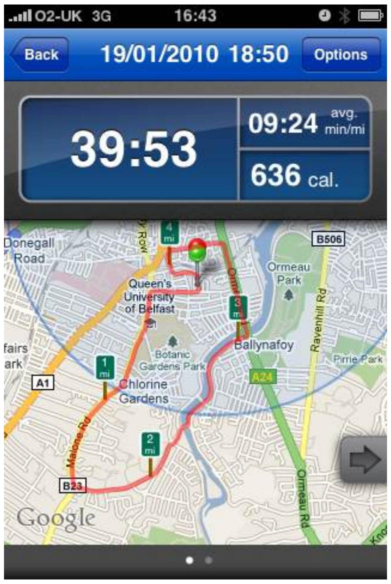


Figure 3.2: The *RunKeeper* interface.

*Nike+ (Nike, 2010)* is a pedometer application that gives runners real-time feedback on their pace and provides a music playlist. Feedback while running is an important concept that I also incorporate into the music and dialog systems of *OnTheRun*. In recent years, *Nike+* has added social, competitive, and tracking features to the product in a manner similar to *RunKeeper*.

*WalkJogRun (AlmostAwesome, 2007)* is one of many online route mapping tools. It allows users to record their runs and search for nearby runs recorded by other users. Users can see information about a run, such as distance and elevation. Services like this empower runners to add variety to their running routes, but *OnTheRun* does this by randomly generating routes using map data.

*RouteA2A (Softabar, 2011)* is a route generator that uses the Google Maps API to build a circuit of specified length. Since the APIs do not allow for specific distances, the app chooses random waypoints in an attempt to create a route that matches the user's desired travel distance. The user can adjust these waypoints repeatedly to improve the fit and download the route data or share it online. *OnTheRun* performs these route building calculations natively on the phone while the player is running.

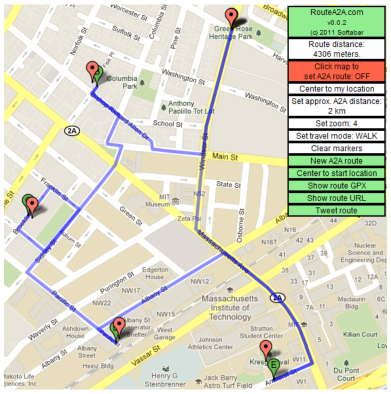

RouteA2A.com
v0.0.2
(c) 2011 Softabar
Route distance:
4306 meters.
Click map to
set A2A route: OFF
Center to my location
Set approx. A2A distance:
2 km
Set zoom: 4
Set travel mode: WALK
Clear markers
New A2A route
Center to start location
Show route GPX
Show route URL
Tweet route

Figure 3.3: *RouteA2A* uses the Google Maps API. Users can drag markers and toggle settings to adjust route generation.

## 3.5 Audio-Only Games

The vast majority of games rely on visuals to create the game experience, but there is a genre of games that are audio-only. These games have been developed for the blind or for the sheer curiosity of novel gameplay. For example, *Papa Sangre* (Somethin'else, 2010), is an iPhone game that puts the player in a world of fantasy and darkness. The player navigates a sequence of levels by tapping his phone's touchscreen to walk. The goal of each level is to reach the exit beacon, while avoiding the grunting and growling enemies out to eat him. To help the player feel more connected, the game presents the sound of the character's footsteps and heavy breathing. The game also requires stereo sound to help orient the player in the game world so that he can move toward the exits and away from the enemies. By contrast, *OnTheRun* uses audio to augment the real world, but it lacks stereo sound because the player's head orientation is not sensed. Instead it relies on narration to explicitly inform the player of the state of the game and uses street names to orient him.

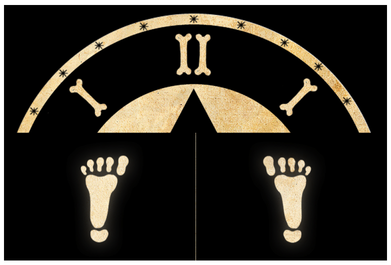


Figure 3.4: The visual interface to *Papa Sangre* only tells the player which direction he is facing.

## 3.6 Alternate Reality Games

Alternate reality games (ARGs) transform the player's normal environment into the game world. Players can receive phone calls from game characters or meet them on the street. ARGs are elaborate and very compelling, but they also require significant resources, preparation and the cooperation of many people. For example, in *Uncle Roy All Around You* (Benford et al., 2004), players use a handheld computer to follow clues and track down Uncle Roy, an elusive character. The clues eventually lead players to Uncle Roy's physical office followed by a parked limousine with a hired actor who asks questions. More recently, *The Witness* (13th Street Universal, 2011) sends players around Berlin, guiding them to particular locations to watch game videos filmed on site. The player's phone detects its location, and plays the video that matches the location. As the game progresses, actors from the videos show up in real life to interact with the players. Because these games are tied to particular locations and require actors, they are very immersive by do not lend themselves to repeatability and personal exercise. *OnTheRun* attempts to automate much of the alternate reality experience, but the cost of this repeatability is the loss of some immersion and interaction.

Figure 3.5: A *Witness* player views a movie in the location that it was film.

## 3.7 Turn-By-Turn Navigation Systems

Since my project makes heavy use of a GPS and turn-by-turn directions, it is worth briefly reviewing the history of voice guided navigation systems, which are ubiquitous today. The standard setup is a combination of voice guidance and a map visualization, and the typical embodiment is a dedicated, dash mounted device or an app running on a smartphone. Since my system is eyes-free, the map visualization is not included, but the voice directions are critical. *Back Seat Driver* (Davis and Schmandt, 1989), a real-time spoken driving directions system, was the first of its kind. *Back Seat Driver* was an audio-only system, freeing the driver to focus his visual attention on the road. Similarly, my system eschews screen interaction so that the runner can pay attention to all the hazards of running on the street. Davis and Schmandt encountered and solved many of the problems I faced in creating the running directions for *OnTheRun*. *Back Seat Driver* used relative directions, detailed descriptions, and local landmarks to guide the human driver to their destination. The system was designed to feel co-present with the driver so that the driver would trust it.

The biggest difference between *Back Seat Driver* and my system is that *Back Seat Driver* does not include street names in its directions. Davis and Schmandt reasoned that not all streets have visible signs and some names can be hard to pronounce. For my system, runners have a better chance of spotting the street sign because they move more slowly than cars. Secondly, text-to-speech technology has improved since 1989, so even if names are pronounced incorrectly, the runner can understand what was said and get the general idea. Some streets are not named at all however, and this was a problem for my system.[2] Lastly, I believe that using actual street names makes the game feel more realistic and provides useful feedback to the player.

[2] My solution was to remove these streets from the database, so that the game will never route the runner down one. This worked but had unintended side effects, which will be discussed in a later section.

# 4 User Experience

*OnTheRun* is a smartphone game that creates an audio experience for a player moving about a city. The game is played without any visual feedback. Instead the player receives audio feedback in the form of music, sounds, pre-recorded voices, and computer generated speech. This audio, combined with the real-world interaction of running and navigating, creates a narrative with the player at the center. To augment the running experience, a run log viewer enables the player to see the routes he has taken, read game dialog, and view supplementary images of the characters and items.

## 4.1 Concept

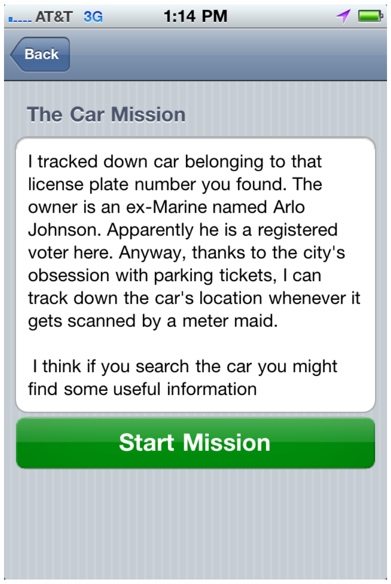
The game is primarily an interactive narrative combined with a navigation system. The game reacts to the player's location and responds appropriately as the player runs around the streets. The game itself is split into a series of missions lasting 5-30 minutes depending on the player's desired run length. Each mission has a unique plot that requires the player to run around, reach destinations, and outrun virtual enemies. During missions the player is guided by a character named Ulysses, who explains the goal of the mission and keeps the player informed of the plot. The player also listens to a computer-generated voice, which provides turn-by-turn directions.

This game relies on a dense map of streets in order to provide the player with descriptions and directions on where to run. Since the game can only output audio, it is useful to engage the player in his surroundings. The game directs the player by using street names, which forces the player to look for street signs. The environment thus provides the visual feedback of the game. The downside of this approach is that the game does not work in open fields, trails, or long roads without side streets: there is not enough descriptive geometry with which to guide the player.[1]

[1] It would be interesting if the player could map out a space first and then play a game: "run past the **stump**, and get to the **park bench**.. you have 60 seconds."

## 4.2 Storyline

Much like an episodic TV show, the missions start without any introduction of the characters or their situation. The general concept is that the player has been framed and is on the run from the authorities. He is also trying to investigate the people who set him up, as they likely framed in order to cover up a larger crime. He is aided by Ulysses, a knowledgeable ally, who speaks to the player over the phone. Neither the player's alleged crime nor Ulysses' motivations are made clear. Before each mission, the player can read about the plot surrounding the mission. The note is in the form of a message from Ulysses.



The Car Mission

I tracked down car belonging to that license plate number you found. The owner is an ex-Marine named Arlo Johnson. Apparently he is a registered voter here. Anyway, thanks to the city's obsession with parking tickets, I can track down the car's location whenever it gets scanned by a meter maid.

I think if you search the car you might find some useful information

Figure 4.1: Each mission has a message from Ulysses that the player can read before running.

## 4.3 Action Sequences

In order to prove the game concept, I made two different missions that are meant to be played in order. Each mission has several different types of interaction. The first playable mission is called "The Car". Ulysses has tracked down a car belonging to a license plate number the player's character previously discovered. Ulysses gives the player the location of the car and and tells the player he only has a few minutes to get to it before the driver comes back. This is an example of a timed-destination objective. As the player runs to the objective, Ulysses reminds him of how much time is left to reach the destination. If the player does not get there in time, he will fail the mission.[2]

[2] The player can replay any mission at any time.

Once the player reaches the car, Ulysses instructs him to break the car's window and steal anything important inside. This is an example of a non-action story sequence. The player is not actually expected to perform this action, but it drives the plot forward. Once the character breaks into the car, an alarm sounds, and Ulysses tells him to head back to the "safe house".[3] The player has 30 seconds to get far enough away from the car. This is an escape sequence, and the next event will trigger once the player has run roughly 100m away from the car.

[3] The safe house is how the game refers to the player's desired end position. Either the player is running back to their starting location, or they entered a custom destination. This could be useful if a player wanted to run home from work.

Part way back to the safe house, the player is warned about an incoming police car. The guidance system routes the player to a hiding spot, and Ulysses warns the player to get there before the cops spot him. This is an example of an avoidance sequence. The player has to stay out of sight of an enemy character. As the player runs, he is given updates of the police car's current location. If the player successfully finds a place to hide, the mission continues. If not, Ulysses will get increasingly concerned until the police car ultimately sees the player, and the mission ends.

Once clear, the player can continue back to the safe house at his own pace. This is a simple no-pressure destination action. The player can take as long as he wants to arrive, and when he does, Ulysses congratulates him on successfully completing the mission. He also says he will investigate the stolen documents and items and figure out what to do next. This sets up the next mission.

The next mission is "The Key". Among the items that the player stole from the car is a key, but the key's purpose is unknown. Ulysses has used the documents to uncover a few potential locations, and he instructs the player to try to unlock them with the key. The player runs to three different locations. After two unsuccessful attempts, and some humorous dialog from Ulysses, the player unlocks a door with the key.

Almost immediately, however, the player is spotted by a guard and must run away. This is a chase sequence; the guard is a virtual pursuer from whom the player must run away. The system guides the player and gives updates of the guard's location. Ulysses also provides feedback of the distance between them, and if the player is too slow, the guard will start yelling threats. The chase sequence ends if the player gets far enough away to lose the guard. As the player gets closer to the safe house, the guard starts slowing down and then abandons the pursuit.

## 4.4 Run Configuration

The plot of each mission is predetermined, but the route the player takes is not. The route depends on the player's current location and desired run length. Before a mission begins, the player selects the desired run length and, optionally, a destination. The game adapts to the player's settings, and a different route will be created each time the player runs (see Figure 4.2). These configurations are done via on-screen interactions. Once the mission starts, the player is directed to destinations. However, the player is free to run wherever he likes, and the system will adapt by continuously updating the shortest path to the current destination. The route planning algorithms are explained in the next chapter.

## 4.5 Dialog and Sound Effects

During the mission, Ulysses provides instructions, explains the plot, and gives feedback on the player's progress. For example, if the player is being chased, Ulysses will warn the player. He will also

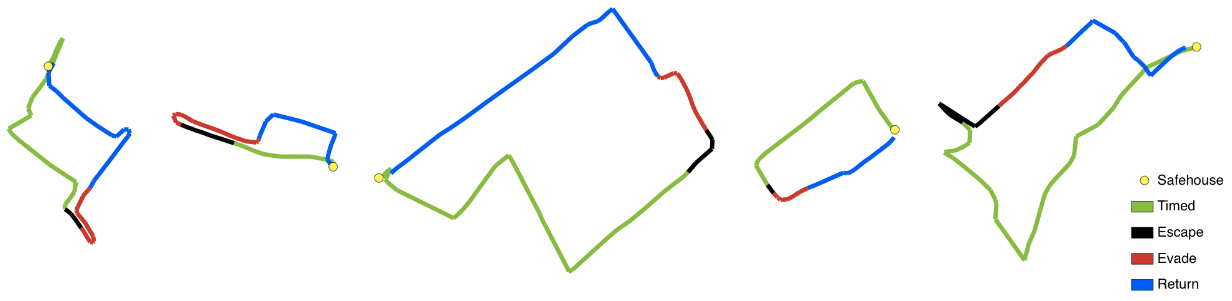


Figure 4.2: These graphs are from different test runs of The Car mission, each tailored to a specific location and length. The routes are drawn at the same scale. The action sequences in each run have the same order, but their relative lengths can vary, as seen by the colored portions of each route.

stress that the player should run faster or that the player is doing a great job in evading detection if this is true. Finally, Ulysses gives the player feedback on how much distance or time remains for the player to reach a destination or if the player is moving in the wrong direction.

Ulysses' voice is pre-recorded and thus has limitations. If the player repeats a mission, he will hear the same recordings again. Also, since the player can be anywhere in world, Ulysses is unable to speak the exact street names. Instead, a computer-generated voice is used to provide turn-by-turn directions to the player, acting as an in-game navigation system that Ulysses has given the player. Though Ulysses refers to it as a GPS unit, it is an advanced one, capable of describing enemy activity as well. Below are excerpts from a sample transcript of a mission[4]. The computer-generated voice is labeled "Computer."

[4] Full transcripts are available in the appendix.

```
Computer: Harvard Street.
Ulysses: The car is parked nearby.
Computer: Your destination is Moore Street toward Broadway.
Ulysses: You don't have much time. The driver will be back soon.
Computer: Go past Columbia street.
Computer: Go past Pine Street.
Ulysses: You have five minutes left.
Computer: Turn left on Windsor Street.
Ulysses: You are halfway there.
Computer: Windsor Street. Turn right on Broadway.
Ulysses: Hold up, I think you are moving the wrong way.
Computer: Turn left on Broadway.
Computer: Broadway, turn right on Moore Street.
```

The computer voice also informs the player of enemy locations.

```
(A faint siren can be heard)
Ulysses: Watch out, there is a cop ahead! Try to avoid him.
```

```
Computer: Your destination is Market Street toward Union Street.
Computer: He just turned left on Columbia St.
(siren is getting louder)
Ulysses: You are gonna get caught! Get off this road!
Computer: Market Street. Continue on Market Street to your destination.
Computer: He just went past Broadway.
Ulysses: All right, you should be safe here.
Computer: He just went past Market Street.
(siren is now getting quieter)
Ulysses: That does it. The coast is clear.
```

The audio from the game is the player's only interaction with the virtual world. He cannot see the enemies and objects described to him. The player interacts by following the directions and looking for the correct streets on which to run.

As a designer it is tempting to use the phone's screen to provide images of the game objects, but I felt that looking at the screen while running was too dangerous.[5] Additionally, one of the major drawback of speech systems is that unless the system is talking, the user can forget it exists (Davis and Schmandt, 1989). If the player goes through long periods of no audio interaction with the game, he may get bored. For both of these reasons, the audio experience needs to be rich.

[5] Though not yet illegal, unlike texting while driving.

I decided to give the system many opportunities to share information with the player. In addition to the turn-by-turn navigation and storyline, Ulysses gives progress updates at periodic distance intervals. The chosen distance intervals are both numerical or proportional.

*Numeric* 1000 meters. 500 meters. 100 meters. 50 meters.

*Proportional* Halfway there. Getting close. Almost there.

Lastly, the volume[6] of sound effects is used to hint at the distance from certain objects. For example, there is a part of the first mission when the player must avoid a police car. The police siren can be heard, and it gets louder as the car gets closer to the player.

[6] I considered using 3D sound, but assumed that it would be inaccurate and distracting because the exact orientation of the player's head is not measured.

## 4.6 Music

Music plays an important part in creating the *OnTheRun* experience. Music is already a popular running accessory. It can motivate the runner and distract him from the pain and boredom of running. Music is also a very important part of gaming. Most major role-playing games have a significant soundtrack. Simple games have a soundtrack that plays independent of the action, but most modern games have a reactive soundtrack that adjusts to the player's actions and progress. For example, *Portal 2*, a recent FPS puzzle/platform game by Valve (2011), has a dynamically generated soundtrack. The music will change just as the player is solving a part of the puzzle, making the player feel like he solved it exactly on cue.

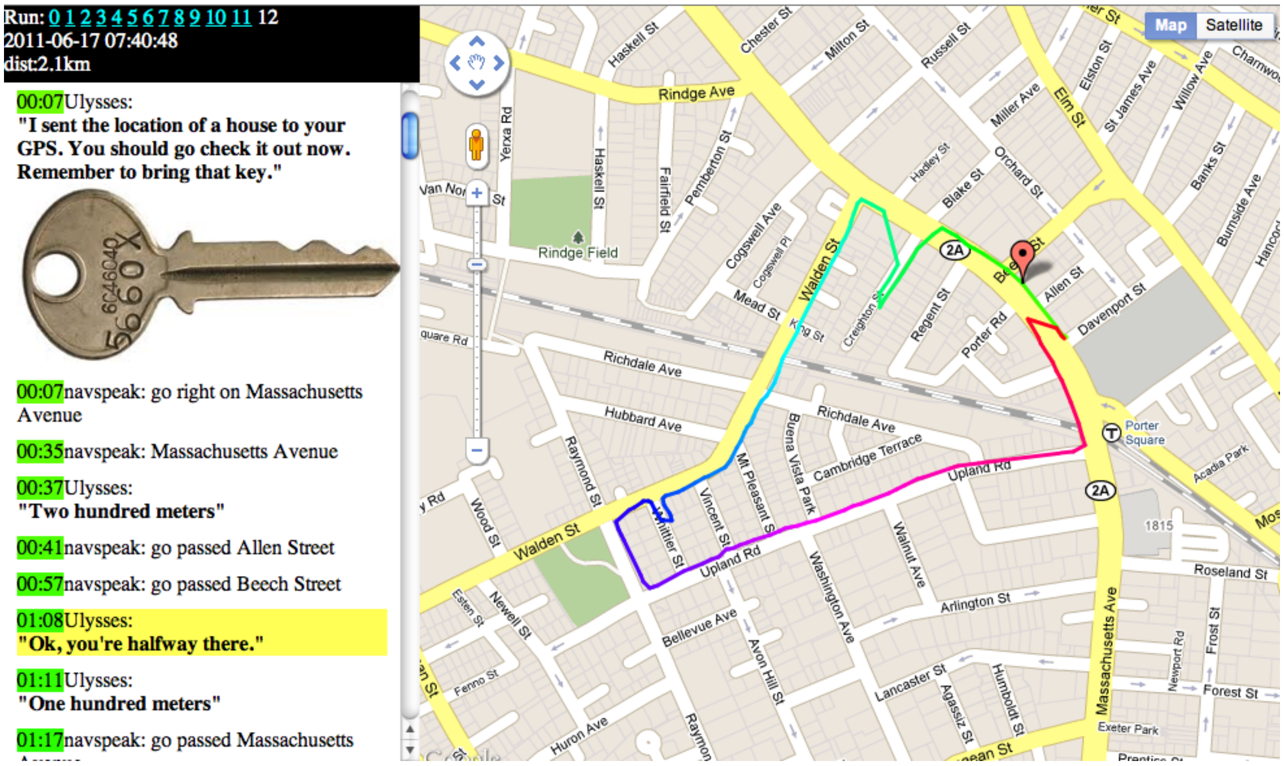
In *OnTheRun*, the music is limited to three separate track loops. The different loops vary in intensity, and the missions switch from loop to loop depending on the current plot point. For example, when the player is running to a destination, the low intensity music plays, but when he is being chased, the high intensity music plays. This simple system gets the job done, although more would be better. A future version of this application would benefit from musical variety.

## 4.7 Post-Run Reflection

After a run, the mission log is uploaded[7] to a server for later analysis, and the player can access this data using a web browser. The player can see a list of the runs he has completed. Selecting a run, the player can view the route taken on a map and read a transcript of the dialog that was spoken. This interface is designed to help the runner reflect on his experiences. The player can also see images of some of the plot elements with which he interacted during the run. Since the run experience is audio only, these images are the first time a player has actually seen these objects. This feature should help the visualization of future missions.

[7] The automatic upload feature is not implemented at this time. I currently have to manually copy the data from the phone to the server.

Players can also use the map to read the street names from their routes and reinforce their memory of their surrounding neighborhood.


Run: 0 1 2 3 4 5 6 7 8 9 10 11 12
2011-06-17 07:40:48
dist:2.1km

00:07Ulysses:
**"I sent the location of a house to your GPS. You should go check it out now. Remember to bring that key."**

00:07navspeak: go right on Massachusetts Avenue

00:35navspeak: Massachusetts Avenue

00:37Ulysses:
**"Two hundred meters"**

00:41navspeak: go passed Allen Street

00:57navspeak: go passed Beech Street

01:08Ulysses:
**"Ok, you're halfway there."**

01:11Ulysses:
**"One hundred meters"**

01:17navspeak: go passed Massachusetts

Figure 4.3: Screenshot of the run viewer web app. The left side is a list of dialog and plot photos. The right side is a map of the player's path as measured by the iPhone's GPS. The player can mouseover individual lines of dialog to see the exact point on the map where that line was spoken.

# 5
# Technical Challenges

The code for this project consists of an iPhone app written in Objective-C and a map server written in Python. There is also a web-based mission log viewer written in Javascript and HTML. A mapping system and turn-by-turn directions algorithms were written from scratch. Street data is from the OpenStreetMap project. The run viewer uses the Google Maps API.

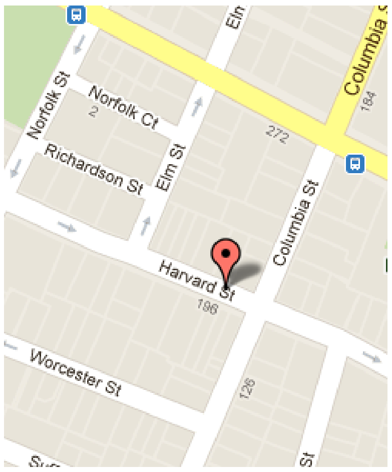


Figure 5.1: A map location can be represented in many different ways. Lat-longs are concise, but a street-centric approach is more workable. (Map ©Google)

## 5.1 Map Representations

The iPhone4's GPS unit calculates its location using standard latitude and longitude coordinates (lat-longs) and has a refresh rate of about one second. While it would be possible to make the game work in lat-long space, I decided to use a different representation that would make it easier to compute directions.

To make the programming easier, I converted the lat-longs to a street-centric representation. Instead of being a coordinate in an X-Y space, the player is represented as a position along a network of nodes and edges. Each node is like an intersection of streets, and each edge is a section of a street between two intersections. This way, instead of saying that the player is at <+42.367062, -71.09813>, the system can say that the player is on Harvard St, past Elm St, facing Columbia St, 20m from the intersection. Using a street centric approach makes it easy to measure distances, compute directions, and move virtual enemies.

The only challenge is that a GPS unit outputs location as a lat-long, and thus a conversion process must be used to transform the player's measured location into a usable representation. The simplest conversion is to snap the lat-long to the nearest street. The system uses the lat-longs of all the street nodes (intersections) to calculate the distance from the player's lat-long to every street in the network. That distance is the error between the street and the player's position.

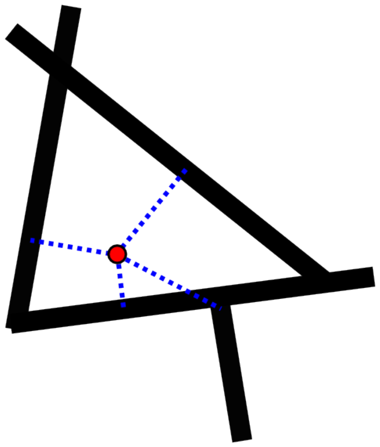


Figure 5.2: The blue dashed lines represent the shortest distance from the red lat-long coordinate to each street. The system uses these distance measurements as part of the localization process.

The street with the lowest distance is likely to be the street that the player is currently on.

Unfortunately, just relying on street distance can cause problems if that distance is less than the average GPS error. If a player is running down a street and passes through an intersection, the system could get confused and think that the player turned at the intersection and started running down a side street. Due to GPS uncertainty, that side street may seem closer than the street the player is actually on. At first I considered looking at old GPS positions to tell if the player was turning. Unfortunately this introduces lag into the system, and I wanted the response time to be as quick as possible.

Instead I take into account the fact that the game is telling the player where to turn. The algorithm assumes the player is following directions and is more responsive to street changes that are directed by the game's guidance system. To accomplish this, a penalty is given to all streets except the current street the player is on and the street onto which the guidance system is telling him to turn. For example, if the player is on Current Street, and the system has told the player to take a left on Next Street, then any street other than Current Street and Next Street gets a score penalty when the system is calculating the player's new location. As a result, the system is very responsive to a player doing the right thing, but it waits 20m or so before telling the player he turned down the wrong street.

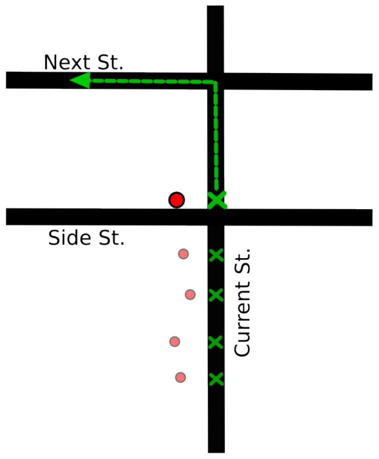


Figure 5.3: The localization system takes into account the instructions for the player and responds quicker to turns it told the player to make. The green arrow is the intended path, the green X is the player's calculated position. The red circles are GPS readings. Even though the latest reading is closest to Side Street, the system still believes the player is on Current Street, as that is where the player should be.

## 5.2 Map Preparation

The map data is from the OpenStreetMap project. All the Cambridge, MA, data was batch downloaded from openstreetmaps.com using their API[1]. OpenStreetMap has two main data types: nodes and ways. A node is a latitude-longitude pair with a unique ID, and a way is a collection of node IDs that represent a street, trail, highway, building, or other map feature. The original download for the Boston area was about 400MB, but purging all the building geometry and redundant street data reduced the download to about 10MB.

[1] Visit http://openstreetmaps.com/api/ for more information on how to do this yourself.

The map data contained roughly 10,000 streets, though not all of them would be acceptable for use in a running game. First, all streets without names were purged, because the turn-by-turn direction system would not be able to describe them. Second, all highways were deleted because they are too dangerous for running. After some initial play testing, all dead-end streets and private ways were removed, because it was uncomfortable and inappropriate to run down someone else's driveway. This made some game-logic algorithms easier to program since I could assume there would never be dead-ends.

These modifications were done automatically using Python scripts to refactor the data. The resulting map data was about 5MB.

One problem with using maps is that those provided by OpenStreetMap are made with vehicle navigation in mind. Many streets are represented as two-lanes, and frequently only one of the lanes will connect to a side street. Vehicle motion is more constrained than pedestrian motion, so many routes that a runner could take are not available to the car. This means that *OnTheRun* will give inefficient running directions in some situations, like having the player continue well past an obvious turn so that the player can make a legal turn to get into the correct lane. I attempted to fix this by hand-editing the map[2], but it proved too time consuming.

[2] It is easy to make an interface for deleting nodes and edges, but moving and reattaching edges requires a more full-featured graph editor.

## 5.3 Map Server

When the player starts a mission, he chooses a desired running distance, and the app contacts a map server to get the map data. The server uses the player's parameters[3] to build a custom subset of the larger map, which is served to the phone.

[3] The parameters are the player's starting position, desired run length, and an optional end position.

The map server works by performing a breadth-first search rooted at the player's current position, acquiring all the nodes that are within the player's desired run length. From this collection of nodes, all the corresponding streets are gathered. This process can create virtual dead ends if some of the nodes in a street are too far away. As a result, the virtual dead ends are iteratively removed from the subset map. The system also automatically increases the run length if the player's desired end position is too far away.[4] Once all the streets and nodes are collected, the server outputs the data as a JSON string.

[4] Runs are limited at 5km. Longer runs require more map data in the phone's memory, and I didn't want to risk low memory problems.

This entire map subset calculation takes the server about a second to compute, and the phone needs about 5-10 seconds to download the data over a 3G connection. Once the phone downloads the map, it converts the data from the compact "nodes and streets" representation to a graph structure with nodes and edges. This structure consumes more memory, as each street gets broken up into multiple edges, but it is more usable. The specific implementation[5] was inspired by Downey (2008).

[5] The graph is represented using nested dictionaries. The top level dictionary has keys for each node in the map. The value for any key is a smaller dictionary with keys for each neighboring node.

## 5.4 Route Builder

Once the phone has the subset map, it builds a route for the current mission the player has selected. Each mission has custom code for choosing destinations, but they all use the player's current position, desired run length, and optional end point as inputs. They also make

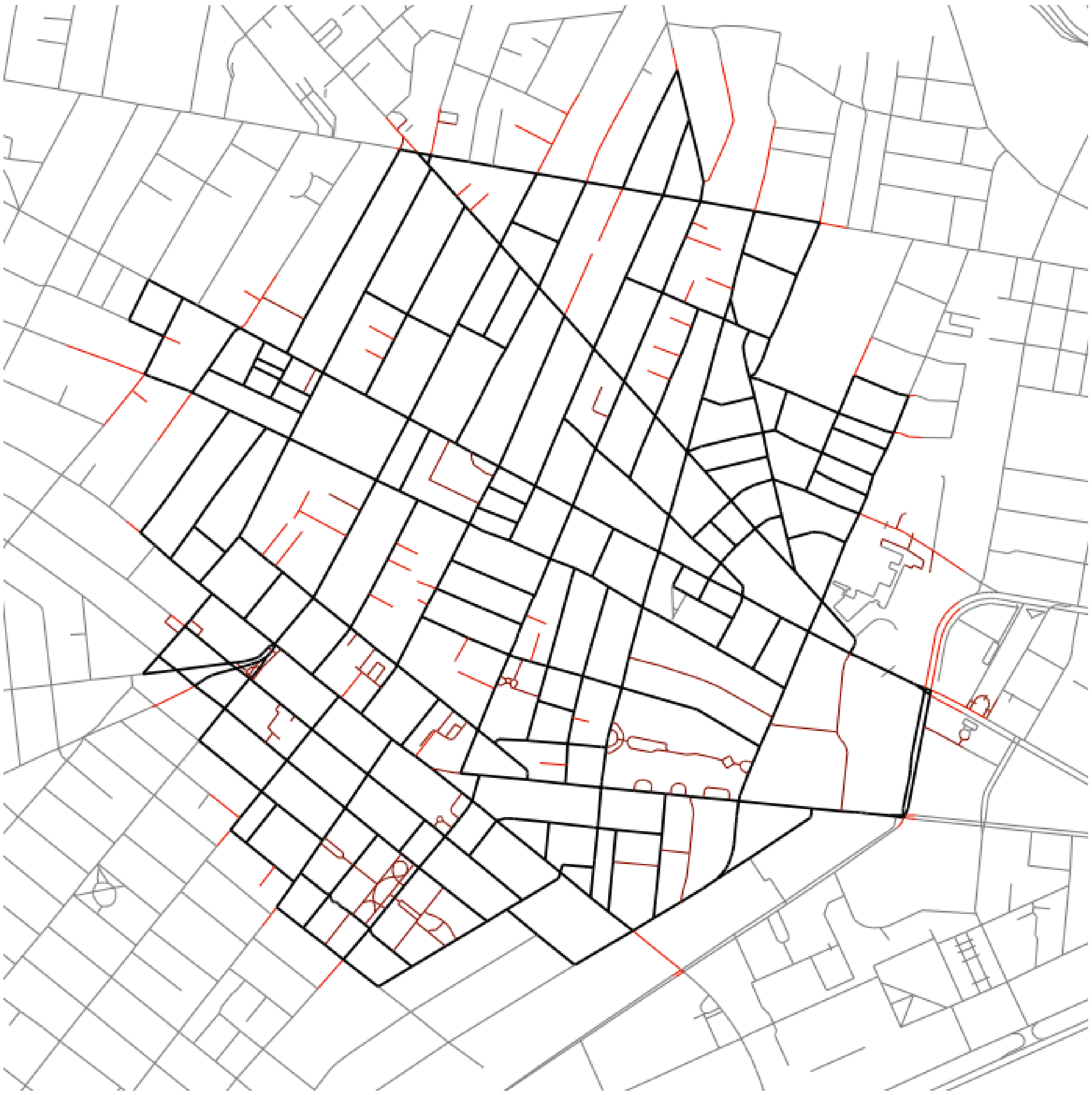


Figure 5.4: The map server processes street data to match a request from the iPhone client. Starting with a center point, unnamed streets (dark red) and streets that are too far away (gray) are removed from the data set. Then all the dead ends (red) are dropped, resulting in a clean and fully connected subset of streets (black).

use of an algorithm that finds points halfway between two other points, given a total distance; it can be thought of as finding the third point of a triangle given two points and the perimeter, except that it works in street-space.

For example, the first mission requires the player to run to a virtual parked car, break in, steal documents, and run away to the safe house. The route builder will place the car at a random location that is half the total run length from both the start and end points.

There are other methods for moving along paths, and these are used for placing sub-goals in the mission. For example the second mission has the player run to several different destinations. The algorithm works by placing the halfway point, and then backtracking along the shortest path to the start point, placing two sub-goals along the way.

## 5.5 Turn-by-Turn Directions

The majority of the player's interaction is with the game's navigation system, which is constantly directing the player to his next destination. These directions are calculated on the phone itself, using the downloaded map data and custom algorithms I wrote for this project. Though making use of the phone's internet connection to request directions from a server was considered, I decided that the game would be more responsive if the directions were calculated on the device itself, without have to rely on a variable-latency 3G data connection to a server.

The direction system uses a modified Dijkstra's algorithm for planning paths and calculating distances between points. The system treats the the map as a symmetric[6] graph, meaning that if node A connects to node B, then node B also connects to A. To find the directions between two points, it performs a breadth-first-search around one point with an optional maximum distance. That point is now considered the root of the resulting "PathSearch" object. This PathSearch can be queried to calculate distances and routes between arbitrary points and the root node. PathSearches can be saved to speed up later calculations involving the same root point. For turn-by-turn directions, the destination is set as the root, and a PathSearch is created once and reused many times. Any time the player position is updated, the system uses the PathSearch to find the best route to the destination and it uses this route to create verbal directions.

[6] Most driving navigation systems use an asymmetric graph in order to handle one-way streets.

The direction system is very simple and merely tells the player what he should be looking for at any given moment. The algorithm starts at the player's location and traverses the current street toward the destination until it encounters a side street. If the side street is on the shortest path, the player is told to turn onto it. If not, the system just tells the player to go past the side street. The result is that the player is always told directions relative to streets that are close by.

Occasionally a player can misunderstand an instruction and run in the opposite direction that the system intends. In these situations, the system detects that the player has made 30m of negative progress and tells the player he is moving the wrong way.

The system can also describe the approximate location of the player's destination. It does this by saying what street the destination is on as well as the next street the player will see when he arrives.

```
Computer:Your destination is on Harvard Street, toward Columbia Street.
```

The process of speaking these directions in conjunction with other dialog is explained in the Audio section.

## 5.6 Dynamic Enemy Positioning and Movement

Both missions have virtual enemies that the player must avoid. In the first mission, the player triggers a car alarm and must avoid an inbound police car. In the second mission, the player stumbles upon an enemy guard and is chased back to the safe house. In these situations, the enemy's initial position is dynamically placed to provide the player with the best experience.

On-demand dynamic placement is a technique I adopted after months of relying on pure random placement. My first attempt would randomly place enemies on the map at the beginning of a mission. As the player ran, the enemies would move randomly as well. The player would be alerted to nearby enemies, and if he got too close, the enemy would chase him. This resulted in an unpredictable experience. Sometimes I would never encounter the enemy, other times he would be inescapable and I would lose. This unreliability was unacceptable, because I knew that most testers would only ever play each level once. That first play had to be perfect.

I struggled for a time, experimenting with different algorithms for enemy starting locations and movement, but ultimately the player's movement and the street layout are too varied and unpredictable; there were too many variables for me to account for and characterize. As a result I developed algorithms that scan the environment and place enemies on-demand in exactly the right spots as needed by the storyline. It feels like a cheat, but it results in a consistently exciting experience.[7]

[7] This consistency affects replayability, but that is a problem for all single-player games.

The dynamic placement is used once in each mission. In the first mission, the player must find a place to hide such that the police car does not intersect his location. This can be tricky depending on the surrounding streets. For example, if the player is running down a long road with no side streets, there is no place to hide and the police car will definitely find him. I didn't want to leave the climactic scene up to chance, so I wrote code to make sure the sequence occurs when the streets can support it. After the player triggers the alarm, he is routed back to the safe house. Once the system detects a side street, the player is alerted to the incoming cop car and is routed to the side street. The cop car is placed far enough away that the player will have enough time to escape given his current pace.

The second mission has the chase sequence with the guard, but when the chase starts, the guard isn't actually on the map yet. Once the player has run 30m in a particular direction, the guard is placed behind him. This way I can ensure that the player doesn't accidentally run directly toward the guard in the stressful moments at the start of the chase. The player is unaware of this, of course, and assumes the guard is behind him the entire time.

## 5.7 Audio Challenges

One of the challenges of creating an "audio only" game is that it can be difficult to present players with all the information they need without overwhelming them. Visual systems display as much information as can fit on a screen simultaneously because the users can focus their attention on a specific piece of information and ignore the rest. This gets more challenging with audio because information must be presented over time rather than space. For example, if the user is a certain distance from a destination, a visual system can continuously display that information in the corner of the screen without bothering the user; the next time the user looks at it, it will be correct. But an audio system must explicitly tell the user the distance, and that can be distracting and annoying. While it is possible to present information in simultaneous audio streams, it can be difficult to parse if many voices are talking at once. If the game said all the information it could, the information would just become noise.

Instead the game is programmed to speak in response to information events:

- The distance to the destination is at a round number.
- There are a round number of minutes remaining on the timer.
- The current turn-by-turn instruction has changed.
- The player has traveled 20m in the wrong direction.

- The player has reached the destination.
- An enemy has turned or passed a street.
- An enemy has gotten 25% closer or farther from the player.

This event-driven approach orders the information in an easily consumable way, but the system must remember what was spoken and under what circumstances. For example, to speak the current distance, the system cannot just wait until the distance is a round number, because the distance could get skipped or repeated. Instead the system has to save the last distance it spoke and wait until the current distance reaches the next useful number range. I wrote reusable class code that makes this process easier, since this structure is used several times per mission. Similarly, the guidance system remembers what the previously spoken direction was. Every time it gets a position update for the player, it recalculates the directions, and it will only speak if the new direction is different from the previously spoken one. When the player is outrunning an enemy, the system reports the enemy's location and then repeats the player's objective. This repetition under pressure adds intensity to the game's action sequences.

# 6 Evaluation

*OnTheRun* was evaluated by six testers who tried a combined total of 16 missions. Five testers were able to complete at least one of the missions, and three had time to try both missions. Tests were conducted serially, allowing me to fix bugs discovered by one tester in time for the next. Log data from each test was also collected and visualized, and three testers were available to review their logs online using the run viewer. Testers were interviewed[1] in person once they had completed their trials, and this chapter is devoted to feedback gathered from those semi-structured interviews.

[1] A list of questions used to conduct the interviews are included in the appendix.

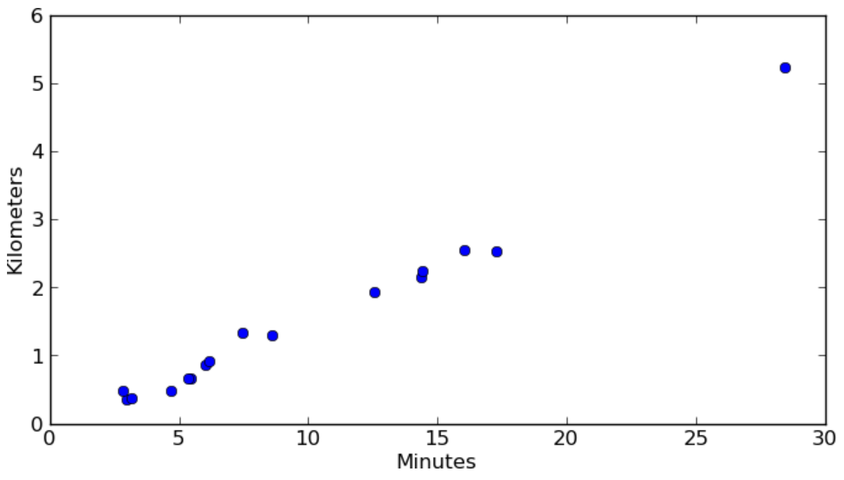


Figure 6.1: This graph compares run time and distance for the 16 different missions. The distance slider allowed players to specify 0.5km-5.0km runs, but the actual distance would vary if the player canceled the mission or ran the wrong way.

## 6.1 Game Mechanics

The majority of the testers enjoyed the action sequences in the second half of each mission.[2] They enjoyed the pressure of being chased, or avoiding a threat. They especially liked how Ulysses would get frantic if the player wasn't moving fast enough.

[2] One tester never made it to the action sequence due to route confusion, and the other mostly ignored the chase sequence.

However, during the police car sequence, the system would announce what street the car was currently on, even when it was far away. Since the players weren't familiar with those streets, they didn't care as much about those announcements. The chase sequence was better because the enemy was following on the same streets as the player. For instance, the player would make a turn, and then a few moments later the system would say that the enemy had just made the same turn. Players recommended that the system announce how far away the enemies were in addition to their locations.

The non-action sequences worked without many problems. Players understood that they were not actually supposed to perform the actions Ulysses asked for (breaking windows, trying locks), but some players weren't sure if they should keep running or not. The players also said that when Ulysses refered to virtual game objects, like houses or cars, they would look nearby for a real object that matched. In one case, Ulysses told the player she had reached the car, but there wasn't a real car nearby, so she ran ahead in order to be next to one.

Most of the players liked the dialog from Ulysses, and two made special note of a particular line of dialog that made them laugh.[3] Players also liked when Ulysses gave them feedback on their performance.

[3] Ulysses gives the player a destination, but once the player arrives, he realizes he made a mistake and sends the player somewhere else. He then admits that he "always gets East and West confused."

Players said that plot was not as dense as it should be, but the players also didn't read the text messages from Ulysses that describe the missions beforehand. For example, in the first mission when the player runs to a parked car, the text explains who owns the car, how he tracked it down, and why the player should go to it. Players just skipped over it and started their missions, and they said they didn't want to stand around reading when they could be running. To make matters worse, the tester who ran 5km felt his run was especially boring, since the story events were spread so far apart. Further discussion of suggested improvements for a future version of *OnTheRun* are included in the next chapter.

## 6.2 Turn-by-Turn directions

The directions from the guidance system added content to the game's sparse dialog, and players enjoyed being told where to go. One player said she actually enjoyed intentionally not following the exact directions in certain situations.[4] However, players universally said that confusing or incorrect directions were their number one frustration with the system. Discussion for improving the localization and direction system is included in the next chapter.

[4] "I liked disobeying the directions. I felt like Jason Bourne having to make my own route."

My hand-coded path planning had a fair share of bugs and recommendations for improvement. Deleting dead ends and unnamed streets from the map was a mistake. In a couple of situations, players mistakenly ran down a street that had been removed from the database, so the system of course thought they were somewhere else and continued to give them bad directions.

When the game first starts, the system only has a few GPS readings. This frequently results in the navigation system thinking the player is facing the opposite way, resulting in wrong directions. In one test run, a player was told to go left when she actually should have gone right. This error, combined with the player's own unfamiliarity with the street layout, caused her to run a half mile down the correct street, but in the wrong direction. She then ran out of time to complete the mission. As a result I modified the system to include warnings about going the wrong way. I also added more types of directions to the system, beyond just turns, so that the player can be sure he is on the right path. For instance, the player will be told to continue past sides streets that are along the path. Later testers had less problems getting where they were supposed to go.

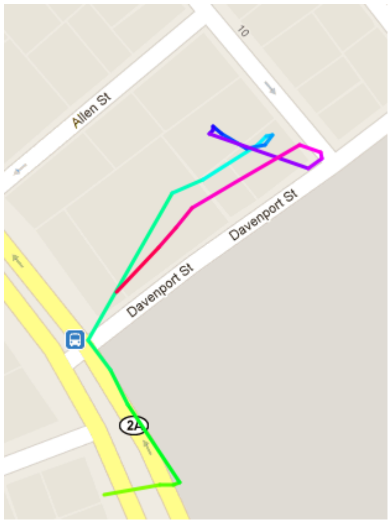


Figure 6.2: Severe GPS inaccuracies caused the system to give confusing directions. This player ran back and forth several times before restarting the mission.

Unfortunately, these fixes introduced new problems. Some players were told they were going the wrong way even when they weren't, and others complained that the system would tell them to continue past a street they had already passed. Frequently, it would tell the player to continue past a street they were currently on, which was annoying but ignorable.

Lastly the players mentioned as feedback that they frequently didn't hear (or couldn't remember) what their destinations were. The system would only say it once, and players said that it was easy to miss the announcement due to street noise. As a result, they would follow the directions without knowing where they were ultimately going.

## 6.3 Route Planning

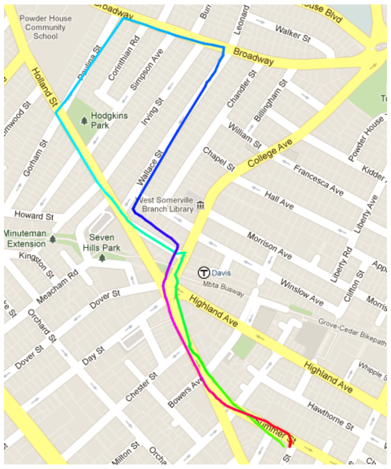


Figure 6.3: This 2.5km run was successfully completed. Notice that the GPS traces mostly line up with the streets on the map.

Generally the random route planning was a success. Most of the players enjoyed taking routes they hadn't run before, saying it was a welcome change to their usual routine of taking the same preplanned path every run. All of the testers who were regular runners said they normally run the same route every time, so having a system that randomized the route for them was interesting. Players said they enjoyed being told where to run. However one runner who normally runs a peaceful route along the river did not like that the system had him run down busy streets.

There was an embarassing bug in the route builder that affected runs with long lengths. Two early testers asked for 4km routes, but the system generated significantly shorter routes.[5] This bug was fixed in time for the last tester, who successfully ran a 5km route.

[5] These routes were of random length varying from 2km to 20m!

## 6.4 Offline Interaction

The run viewer was available in time for three different testers to try it. All three reacted positively to seeing the map of their route, and they said that it was fun to read the dialog again. One tester said it made him feel more accomplished. Even the player who had GPS problems and got confused said it was reassuring to see that the failed mission was not entirely her mistake: the system thought she was on a completely different street.[6] Another player said that seeing his route on the map showed that the GPS was more accurate than he expected. This might be more evidence that the game should do a better job of describing its knowledge of a player's location. The pictures interspersed in the dialog received mix reviews. One tester said he ignored them, while another said she enjoyed seeing them.

[6] GPS problems are very hard to diagnose in the system. My algorithm could be wrong, the GPS might have an error, the signal may be blocked by tall buildings, or there might not be enough satellites in the sky at that moment.

# 7
# Future Work

The nature of my work thus far is exploratory: I propose a specific design for a location-based, single-player, exercise game and demonstrated the feasibility of my prototype. In my design process, I identified a number of ways to improve both the technical aspects of the system and the design of experience itself and this chapter is devoted to sharing these suggested improvements. Furthermore, although I showed that people enjoyed using the system and the system worked, I will also introduce some candidate hypotheses and experimental protocols for future research.

## 7.1 The localization model

The current model takes into account the phone's latest GPS reading. While I worked hard to develop a model that was responsive given only GPS input, a future model that leverages the phone's existing compass and accelerometer would greatly improve the system's accuracy.[1] The compass would help the system identify turns better, and knowing absolute direction would help confirm that the player is on the correct street. The accelerometer would aid in player speed estimation as the system should be able to differentiate between walking, jogging, running and stopping. A speed estimate would help the system predict the player's current location instead of relying on an estimate that might be a second or two old. Players complained that the system would tell them to pass streets they had already passed, or would occasionly take a few seconds to correctly identify a correct turn; this can happen when the GPS suddenly stops updating for a few moments because the system is unable to do any time-based location prediction. By predicting location, GPS dropouts wouldn't be as problematic. Having a speed estimate is critical to that prediction.

[1] These additional sensors readings could also add features to the game.

If the phone can handle the additional processing required, it could be useful to implement a particle filter to predict the player's location. Particle filters work well with non-linear data and could incorporate information like street distances and directions.

## 7.2 Music

The current music in the game is three 40-second loops of different intensity, and ultimately the player is going to get pretty bored with hearing the same two minutes of sound. A variety of tracks would be good. It could also be cool to allow the player to choose his own music, though this might clash with the mood of the game. It could be interesting to experiment with changing the tempo of the song, or automatically selecting songs that match the player's pace.

## 7.3 Spoken directions and descriptions

Since I used exact street names, I did not delve into more descriptive directions employed in *Back Seat Driver*, which would describe street landmarks, intersection geometry, and when the driver should expect to turn. It also included a "What Now" button that users could push to hear what they should do right now, in case they forgot or were uncertain. My system simply says the next street the player should pass or turn onto, and it does not repeat itself.

A future version would benefit from more descriptive directions and timely reminders of what the players should do. One tester recommended that the system provide compass directions in order for the runner to get his bearings better. For example: "Head north on Harvard Street". Another player recommended that the system tell you where it thinks you are, so that you know when to trust the directions. Approximate street numbers would also help players know where a destination is along a street. Lastly, there should be a button to repeat the address of your destination, in case the player does not hear it or can't remember.

## 7.4 Story and Missions

This was just a feasibility demo, but a full game would obviously need more missions and a better storyline. One player suggested that Ulysses should speak the mission description text instead of forcing the player to read it. This would make the beginning of each run more interesting. Another player wanted more feedback on how well he was running because he said it felt good to hear it. Also the system should include more sub-goals when the player requests a longer run, so that the story doesn't get spread too thin.

One tester suggested an interesting new game mechanic. Ulysses could tell the players to stop running and "blend in" to their surroundings in order to lose a pursuer. He might say that the player should slow to a walk or even stop and "pretend to tie their shoe." The system could use the phone's accelerometer to confirm that the player is roughly following orders. The game can be more immersive if it has a variety of interactions.

Screen interactions were explicitly minimized for this project, but one tester suggested that solving on-screen puzzles part way through missions might be fun. This could be an area of exploration, though there was a similar mechanic in the game PiNiZoRo.

## 7.5 Mission Designer

I struggled with *OnTheRun* because there are so many different aspects to game making, and there was significant tension between what I wanted to create and what I was able to do given my ability and time. In a professional video game studio, games are created by teams, and not everyone on a team is a programmer. Most members are actually designers who create game content: story, art, sound, levels, puzzle logic, etc. The job of the programmer is to create tools for the other team members to use to create game content. One of the first things that gets made is a level design tool. The programmer creates an application that can be used by non-programmers to generate all the levels in the game.

In *OnTheRun*, I custom wrote the code for two different missions. An ideal future version of *OnTheRun* would have tens or hundreds of missions, and it would be tedious to program each mission individually. I already employ subclassing to reuse code that is common between my two missions, but an actual mission design tool will be an important part of a full version of this game.

Early in the development of the project, I attempted to create a level editor and logic system for the game so that I wouldn't have to code everything natively in verbose Objective-C. This was a mistake because I didn't yet know the variety of functionality I would need to support. My initial system supported location-based and time-based events. There were lists of active and inactive events. Active events would trigger dialog if their conditions were met and would also activate or deactivate other events. I found this system to be expressive, but dealing with turn-by-turn directions and virtual enemies quickly overwhelmed it. Now that I have successfully made an example game, I think I could develop a mission designer to support my needs, but it would be complex.

The tool would be similar to a traditional videogame map editor. However, unlike most videogames, the mission might get run on any street and thus cannot be fully specified in advance. Instead a designer must draw a simplified version of the game world, and a placement system will map the game world onto the player's surroundings at the start of a mission. The designer places the safe house location, intermediate destinations and dialog points. He creates events so that when the player reaches a specified point, it triggers the dialog to be spoken and activates the next events.

Dealing with virtual enemies is tricky because their behavior can be difficult to generalize. The enemies in my game moved toward destinations or followed the player. In the future they might also need to avoid the player or move at a non-constant speed. The trade off in creating custom tools is between speed and power, because making it easy to do certain actions makes it harder to do others.

## 7.6 Player-specified Running Speed and Difficulty

In the current implementation, players cannot specify how fast they would like to run. As a result, the game must support a wider range of speeds. This means the game might be too easy for people who run fast or too hard for people who jog slowly. It would be better to support three different modes: walking, jogging, and running.

This would introduce a new challenge for mission designers since the game would need to be playable and fun no matter what pace the player runs. It may not be fun to have a chase scene in which the pursuer can be avoided simply by walking. In these cases, it might make sense to treat running speed as a difficulty factor, and adapt the content of the missions to the different difficulties. In easy (walk) mode, the player may never get chased, but in harder modes he does. This type of difficulty-dependent plot is common in modern games.

## 7.7 Viewable map

Players would like to be able to see a map of the route right before the run, and they want to be able to stop and check it during the run in case something goes wrong with the direction system. Visually seeing the destination and where the system thinks the player is can help eliminate confusion.

## 7.8 Better pedestrian maps

The current system does not handle complex geometry such as two lane streets. In a future version, I would either write algorithms that acknowledge what side of the street the player is on, or edit the map to convert all two-lane streets to a single lane. One tester recommended adding crosswalk data to the map. Another tester suggested augmenting the map data with a notion of the "runnability" of a street, so that busy streets are avoided as much as possible. Building off work by Priedhorsky et al. (2007), *OnTheRun* could maintain a *geowiki* of user maintained runnable streets. Users could rate vehicular roads on a runnability scale and even add new paths and trails that are not in traditional driving databases.

## 7.9 Local landmarks

Two players suggested that landmarks could improve the quality of the directions and make the game seem more engaged with reality. Again this was something included in *Back Seat Driver*, and *OnTheRun* would benefit from adopting it. These landmarks could be mined from existing data sources or provided by the player when the game is first configured. The game can interact with the player by using phrases like "Your contact will be waiting for you at a bus stop on Main Street" or "Turn right after the book store." The route planning algorithm can select local landmarks that best match the plot of the mission, and then place enemies and destinations appropriately.

## 7.10 Run Log Viewer

In a future version, the log viewer should receive data automatically from the phone after a run. The viewer should also be part of a stand-alone web site, where players create accounts, manage their data, and prepare for new runs. One tester suggested including more local pictures in the viewer so that she could refamiliarize herself with the areas she saw on the run. It could be possible to incorporate Google StreetView imagery, or to have the players take photos with their phone at key plot points and show these photos to them later.

## 7.11 Evaluation Structure

The evaluation goal for this project was to see if the system worked and if people enjoyed using it. With a more robust system, more runners, and a longer testing period, it could be possible to investigate more substantive hypotheses: When using the system, do runners run longer than usual? Do they run more often? Do they enjoy their runs more? To answer these questions, we must compare runner performance before, during, and after exposure to the system.

Testers should be given a smartphone with an app that tracks their running statistics and provides minimal features beyond that of existing exercise apps like *Nike+* or *Runkeeper*. The app should record the runners' routes in real-time, capturing location and speed information. After each run, the app should prompt the player to quantify their enjoyment of the run and their perception of their performance. After using that app for a month in order to gather a baseline performance, testers will be given the complete game app to use for another month. After the testers play the game for a month, it will be replaced with the tracking app again for a final month to study any lasting performance effects from playing the game. The run data combined with the submitted ratings will quantify the impact of the system on player performance.

# 8 Conclusion

This work shows that it is possible to create an engaging single-player gaming experience beyond the confines of the living room. Speech, music, and sound effects alone are able to create a fun game, even if the only interaction is large scale movement in players' surroundings. *OnTheRun* is a game that can be played while running because its user interface makes it possible to interact with the game while still keeping the players' eyes on the road. As mobile phones become more ubiquitous and sensor-filled, games like this will be commonplace. A project like *OnTheRun* provides a demonstration of the potential for interactive, entertainment-oriented games to escape the bounds of the videoscreen and explode into the real world.

The framework introduced by *OnTheRun* has the potential to provide additional motivation and performance benefits to an exercise routine. With enough resources to develop sufficient content, fine-tune player engagement and calibrate system performance, this potential could be realized as a complete product, and the usage of that product could be studied more formally than was possible within the scope of this work.

# 9 Bibliography

13th Street Universal (2011). The witness. Website. `http://www.youtube.com/watch?v=Yis6is8v9jA`.

AlmostAwesome (2007). WalkJogRun. Website. `http://walkjogrun.net`.

Benford, S., Flintham, M., Drozd, A., Anastasi, R., Rowland, D., Tandavanitj, N., Adams, M., Row-Farr, J., Oldroyd, A., and Sutton, J. (2004). Uncle Roy All Around You: Implicating the City in a Location-Based Performance. In *ACM Advanced Computer Entertainment*. ACM Press.

Caillois, R. and Barash, M. (2001). *Man, play, and games*. University of Illinois Press.

Campbell, T., Ngo, B., and Fogarty, J. (2008). Game design principles in everyday fitness applications. In *CSCW '08: Proceedings of the 2008 ACM conference on Computer supported cooperative work*, pages 249–252, New York, NY, USA. ACM.

Davis, J. and Schmandt, C. (1989). The back seat driver: Real-time spoken driving directions.

Deci, E. and Ryan, R. (1985). Intrinsic motivation and self-determination in human behavior. *books.google.com*.

Downey, A. (2008). Computational modeling and complexity science. pages 3–6.

Flintham, M., Benford, S., Anastasi, R., Hemmings, T., Crabtree, A., Greenhalgh, C., Tandavanitj, N., Adams, M., and Row-Farr, J. (2003). Where on-line meets on the streets: experiences with mobile mixed reality games. pages 569–576.

Fujiki, Y., Kazakos, K., Puri, C., Buddharaju, P., Pavlidis, I., and Levine, J. (2008). Neat-o-games: blending physical activity and fun in the daily routine. *Comput. Entertain.*, 6(2):1–22.

Hielscher, J. and Heitlager, J. (2006). Wanderer–location independent gps game. *PerGames seminar*.

Konami (1999). Dance Dance Revolution. Website. `http://www.konami.com/games/ddr`.

Koster, R. (2005). *A theory of fun for game design*. Paraglyph Series. Paraglyph Press.

Lin, J., Mamykina, L., Lindtner, S., Delajoux, G., and Strub, H. (2006). Fish'n'steps: Encouraging physical activity with an interactive computer game. In Dourish, P. and Friday, A., editors, *UbiComp 2006: Ubiquitous Computing*, volume 4206 of *Lecture Notes in Computer Science*, pages 261–278. Springer Berlin / Heidelberg.

Microsoft (2011). Fun that's good for you - xbox.com. Website. `http://www.xbox.com/en-US/Kinect/healthyfun`.

Nike (2010). Nike+. Website. `http://nikeplus.com`.

Nintendo (2007). Fitness game for Nintendo Wii - Wii Fit Plus. Website. `http://wiifit.com/`.

PerBlue (2011). Parallel Kingdom. Website. `http://www.parallelkingdom.com/`.

Priedhorsky, R., Jordan, B., and Terveen, L. (2007). How a personalized geowiki can help bicyclists share information more effectively. In *Proceedings of the 2007 international symposium on Wikis*, WikiSym '07, pages 93–98, New York, NY, USA. ACM.

Runkeeper (2010). Running app and fitness community. Website. `http://runkeeper.com`.

Softabar (2011). RouteA2A - When You Want To Go To Where You Already Are. Website. `http://routea2a.com`.

Somethin'else (2010). Papa sangre. Website. `http://www.papasangre.com/`.

Stanley, K., Livingston, I., Bandurka, A., Kapiszka, R., and Mandryk, R. (2010). Pinizoro: A gps-based exercise game for families.

Valve (2011). Portal 2. Website. `http://www.thinkwithportals.com/`.

Whitehead, J. R., on Physical Fitness, P. C., and (U.S.), S. (1993). *Physical activity and intrinsic motivation [electronic resource] / James R. Whitehead*. President's Council on Physical Fitness and Sports, Washington, DC.

# *A* *Mission Transcripts*

Included are one example transcript from each mission.

## A.1 The Car Mission (0.8km)

*Computer* Sidney Street

*Ulysses* Alright, the car is parked nearby

*Computer* Your destination is Pearl Street toward Erie Street.

*Ulysses* You don't have much time. The driver will be back soon.

*Computer* Go right on Erie Street

*Ulysses* Four minutes.

*Computer* Erie Street

*Computer* Go past Sidney Street

*Computer* Go past Brookline Street

*Ulysses* Ok, you're halfway there.

*Ulysses* Three minutes.

*Computer* Go left on Pearl Street

*Ulysses* Two hundred meters

*Ulysses* You're getting close.

*Ulysses* One hundred meters

*Ulysses* Just two minutes.

*Computer* Pearl Street

*Computer* Go past Erie Street

*Ulysses* That's the car right there. We don't have much time, so we'll have to improvise: smash the window and steal anything you can find inside.

*Ulysses* Great. Now you've got about thirty seconds to get the hell out of there.

*Computer* Your destination is Sidney Street toward Emily Street.

*Ulysses* The police will be checking out that alarm, so be sure to get out of there.

*Computer* Go right on Valentine Street.

*Ulysses* I've detected an incoming police car. It's coming straight for you. You have to get off its course.

*Computer* Police activity on Sidney Street.

*Computer* Your destination is Brookline Street toward Emily Street.

*Computer* Go right on Valentine Street.

*Computer* He just turned onto Erie Street.

*Computer* Go right on Valentine Street.

*Computer* Valentine Street.

*Computer* Go left on Brookline Street.

*Computer* He just went past Brookline Street.

*Ulysses* Ok. You should be safe here for now.

*Computer* He just turned onto Pearl Street.

*Ulysses* That does it. The coast is clear.

*Computer* Your destination is Sidney Street toward Emily Street.

*Computer* Go left on Brookline Street.

*Computer* Brookline Street

*Computer* Go right on Emily Street.

*Computer* Emily Street.

*Computer* Brookline Street.

*Computer* Go left on Emily Street.

*Ulysses* Two hundred meters.

*Computer* Emily Street.

*Computer* Go right on Sidney Street.

*Ulysses* Ok, you're halfway there.

*Ulysses* One hundred meters.

*Ulysses* You're getting close.

*Ulysses* Fifty meters.

*Ulysses* Alright, I think we can call this a successful mission. Send me some photos of the stuff you stole and I will contact you again when I know what to do next. Stay safe.

## A.2 The Key Mission (1.3km)

*Computer* Elm Street

*Ulysses* From what I can tell from the files they own a bunch of properties in the area, and I have a hunch that this key will work with one of them. The first house is nearby.

*Computer* Harvard Street.

*Computer* Your destination is Elm Street toward Market Street.

*Ulysses* I sent the location of a house to your GPS. You should go check it out now. Remember to bring that key.

*Computer* Go right on Elm Street.

*Computer* Go left on Elm Street.

*Computer* Elm Street.

*Computer* Go past Richardson Street.

*Computer* Go past Norfolk Court.

*Computer* Go past Broadway.

*Ulysses* One hundred meters.

*Computer* Go past Elm Street.

*Ulysses* Fifty meters.

*Ulysses* Ok, this is the first location. Try the key.

*Ulysses* Did that work?

*Ulysses* Alright. This place is no good.

*Ulysses* Hmm. Ok let's try another place.

*Computer* Your destination is Elm Street toward Hampshire Street.

*Computer* Go past Market Street.

*Computer* Go past Hampshire Street.

*Ulysses* One hundred meters.

*Ulysses* Fifty meters.

*Ulysses* There should be a storage unit around here somewhere. I think? Oh... my mistake. Keep moving.

*Computer* Your destination is Hampshire Street toward Murdock Street.

*Ulysses* I always get East and West confused... keep moving.

*Ulysses* Two hundred meters.

*Computer* Go left on Hampshire Street.

*Computer* Hampshire Street.

*Computer* Go past Norfolk Street.

*Ulysses* Ok, you're halfway there.

*Computer* Norfolk Street.

*Computer* Go left on Hampshire Street.

*Ulysses* One hundred meters.

*Computer* Hampshire Street.

*Computer* Go past Tremont Street.

*Ulysses* Fifty meters.

*Ulysses* You're getting close.

*Ulysses* Ok this is it. Try the key.

*Ulysses* Huh, that's interesting... Do you see anything inside?

*Ulysses* Oh shit... I'm picking up a security guard's radio signal. RUN! Get to the safehouse.

*Ulysses* Hold up. I think you're moving the wrong way.

*Computer* Go past Murdock Street.

*Enemy Guard* Hey you! Get out of there!

*Ulysses* He's still behind you...

*Ulysses* Hold up. I think you're moving the wrong way.

*Computer* Go past Murdock Street.

*Ulysses* Step up the pace. He's gaining on you.

*Computer* He just went past Murdock Street.

*Computer* Prospect Street.

*Ulysses* He's still there.

*Computer* Go past Gardner Road.

*Computer* He just turned onto Prospect Street.

*Computer* Go past Gardner Road.

*Ulysses* He's closing in on you!

*Computer* Go past Saint Mary Road.

*Computer* Go left on Broadway.

*Computer* He just went past Gardner Road.

*Ulysses* He's getting close.

*Computer* Go left on Broadway.

*Computer* He just went past Saint Mary Road.

*Ulysses* Five hundred meters.

*Computer* Go left on Broadway.

*Ulysses* You realize that he's not going to stop, right? You have to outrun him.

*Enemy Guard* You can't outrun ME!

*Ulysses* Keep going, he's still behind you.

*Computer* Broadway.

*Computer* Go past Scouting Way.

*Computer* He just turned onto Broadway.

*Computer* Scouting Way.

*Computer* Go left on Harvard Street.

*Computer* Broadway.

*Computer* Go past Tremont Street.

*Computer* He just went past Scouting Way.

*Ulysses* He's gaining on you.

*Ulysses* He is RIGHT behind you!

*Computer* Go past Tremont Street.

*Ulysses* Ok, you're halfway there.

*Ulysses* You're losing him.

*Computer* Go past Norfolk Street.

*Ulysses* He's getting close!

*Computer* He just went past Tremont Street.

*Computer* Go past Norfolk Street.

*Ulysses* Two hundred meters.

*Computer* Go right on Elm Street.

*Computer* He just went past Norfolk Street.

*Computer* Go right on Elm Street.

*Ulysses* He's getting close!

*Ulysses* You're getting close.

*Ulysses* Don't stop now, you're gonna get us both caught!!

*Computer* Elm Street

*Ulysses* One hundred meters.

*Computer* Go past Norfolk Court.

*Computer* Go past Richardson Street.

*Ulysses* You're losing him.

*Computer* He just turned onto Elm Street.

*Ulysses* He's slowing down, but this ain't over yet!

*Ulysses* You're almost there.

*Ulysses* Fifty meters.

*Computer* Go past Richardson Street.

*Ulysses* He's still behind you...

*Ulysses* Well that didn't go as planned. They are likely going to get rid of any evidence in that house, but I can do some research and see what's so special about that place. I'll talk to you soon.

# *B*
# *Interview Questions*

To allow for variations in tester feedback and communications style, each interview was unique, but I typically pulled from a common set of questions.

*Background*

- What games do you play (video games, board games, team sports)?
- Do you own a smartphone?
- Have you used turn-by-turn GPS navigation before (like in a car)?

*Exercise and Running Habits*

- Do you plan a route when you go running? Do you run the same route every time?
- Do you listen to music while running? Do you change your playlist? Ever listen to an audiobook while running?
- What type of other exercise to you do (running, gym, weightlifting)?
- Do you ever get bored while you run?

*Game Mechanics*

- Did you feel like a fugitive?
- Did the app make you work harder in order to avoid capture?
- When you were being chased, did you ever look back to see how far away the bad guy was?
- Did you ever look at objects in the real world (cars, houses, people) and imagine they were part of the game?

- What parts were fun?
- What parts were bad or frustrating?
- Did you ever repeat a mission? Did it bother you that it was exactly the same?
- Did it bother you that the things described in the mission (cars, houses etc) didn't actually exist? Would you prefer to run to real locations? For example, "head to the CVS on the corner of Church Street and Mass Ave", or "meet your contact at the bus stop nearby"?
- Did you notice that locations would change depending on where you started the mission? Did you care?
- Were the missions interesting? Was there enough action? Would you have preferred more dialog from Ulysses?

### *Route Planning and Directions*

- Did you like having turn-by-turn directions?
- Did you like not knowing exactly where you were going?
- Would you prefer to look at a map of your route on the phone at the beginning of a mission?
- Would you liked to see your route ahead of time on a computer? Even if that meant you had to start the mission from a particular location?
- Were you ever lost and wished you could just look at a map?
- Did the game ever tell you to turn, forcing you to cross a busy street?
- Were you ever waiting at a stoplight while the game was telling you to keep going?
- Were you ever annoyed with directions because you knew how to get there?
- Did you try to set a destination? Did it send you to the wrong place? Did the inefficient route bother you?

*Run Log Viewer*

- Is this useful at all?
- Does it help you remember the experience?
- Do the "story" photos do anything for you?
- Can you see a difference between where the system thought you went and where you actually ran?
- What else would you like to see in the run viewer?
- What exactly did you try? Did you do mouseovers? Did you view other runs?

*Improvements*

- Would you like a freeform app that didn't care what streets you ran on, only how far and how fast?
- What if instead of tailoring the mission to your current location, you had to go to a particular location in order to start a mission? This would enable the game world to be static and consistent.
- Describe any problems you had. Configuring missions, following directions, wearing the device, etc.
- What stats would you like to see at the end of a mission (time, speed, route taken)?
- Would like to be able to pause the game to wait for stoplights or catch your breath?
- Would you like to be contacted by Ulysses between runs, such as during the day?
- Would the game be better with a more in-depth story?
- Would you prefer a game that focused on scoring points and providing quantitative assessment of your runs? This would allow you to compare performance between runs.