Enhancing Game-Based Learning with Citizen Science Concepts

by

Nayoung Lee

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author .....
Department of Electrical Engineering and Computer Science
May 24, 2019

Certified by .....
Eric Klopfer
Director, MIT Scheller Teacher Education Program
Thesis Supervisor

Accepted by .....
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Enhancing Game-Based Learning with Citizen Science Concepts

by

Nayoung Lee

Submitted to the Department of Electrical Engineering and Computer Science
on May 24, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

TaleBlazer is a location-based augmented reality platform for educational mobile games. Currently, typical TaleBlazer games enhance passive learning by engaging players in a fictional narrative, but the platform lacks the capabilities to promote active observation of the real world in its games. On the other hand, there have been numerous past projects and initiatives that note the positive impact of citizen science on participants' curiosity and observation about the world around them. For this reason, I have worked to incorporate citizen science concepts and themes into TaleBlazer. Many of the concepts that are central to citizen science focus on giving participants the ability to collect information; however, due to limitations of the TaleBlazer platform, there is no easy way for players to input information into the games. In order to improve TaleBlazer in respect to this limitation, I have added features that allow players to input information into TaleBlazer games. I have implemented two new types of agents (in-game entities): an agent that allows players to tag their current location as a point of interest, which players can review later in the game, and an agent that can use a player's input to impact the game. I have also implemented a form editor on the online web interface that allows game developers to customize the types of input that plays can submit. These new features not only enhance learning, but also open the door for games that can build on top of player-generated content instead of relying solely on game developer-generated content.

Thesis Supervisor: Eric Klopfer
Title: Director, MIT Scheller Teacher Education Program

Acknowledgments

I'd like to thank Eric Klopfer, the TaleBlazer team, and the entire STEP Lab for giving me the opportunity to work on this project and for the friendly and encouraging work environment.

I'd like to thank Lisa Stump and Judy Perry for all their technical and managerial support, for their wonderful ideas, for their constructive feedback, and for encouraging and empowering me to express my ideas and thoughts. I'd also like to thank my fellow overlapping students in the TaleBlazer team for their support, ideas, and feedback as well: Carlos Henríquez, Sneha Ramachandran, Nisha Devasia, Angela Lin. I'd also like to thank Brandon Hanks for his help and support in wrapping up the project.

I'd also like to thank Sophie Russo for her continued support and feedback throughout the year. I'd also like to thank all my friends who have helped me make the most of this last year at MIT.

Lastly, I'd like to thank my parents and my brother for their support, love, and encouragement all my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The MIT Scheller Teacher Education Program (STEP) Lab focuses on improving education with technology. One of their active projects is TaleBlazer, a location-based augmented reality (AR) platform for educational mobile games. TaleBlazer launched in 2013 and is currently being used as an educational tool in schools and other informal settings (e.g., museums). It is also continuously being improved as new features are added to enhance the platform. TaleBlazer's homepage is shown in Figure 1-1, and it highlights how TaleBlazer's platform can be used for both making games and playing games.

I have been working with the TaleBlazer group for my Masters of Engineering (MEng) project. My project spanned from June 2018 to the end of May 2019. Using the current TaleBlazer platform, it was difficult to incorporate citizen science ideas and themes into games, because there was no easy way to add player-generated content to the games. To solve this problem, I developed features that allow players to input and review information in-game. These features will also enhance a user's learning about the act of research and observation about the world around them. Throughout the rest of this paper, I discuss the motivation for this work and explain the work I have done.

Chapters 2 and 3 provide some background information that is helpful in understanding the scope of the project. Chapter 2 talks about TaleBlazer's current users, relevant game concepts, and typical game mechanics. Chapter 3 explains some further

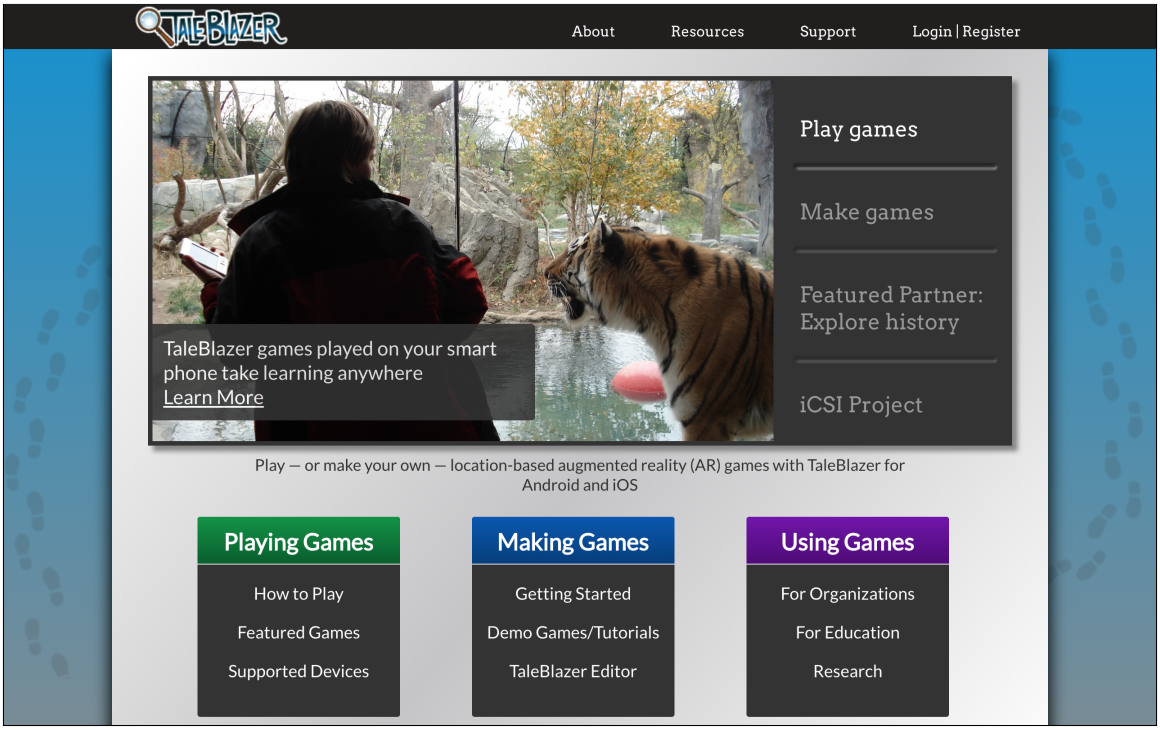


Figure 1-1: TaleBlazer's homepage.

background related to the citizen science field, including some significant projects and an interview with a user of a citizen science app. Chapters 4 and 5 explain overarching design goals. Chapter 4 ties together both of the previous background chapters by further explaining the motivation behind incorporating citizen science into the TaleBlazer platform. Chapter 5 then illustrates the desired use cases and lays out some design principles that guided the project. Chapter 6 then outlines each of the new features added to TaleBlazer, how they were designed, how they each work, and how they blend into the existing platform. Chapter 7 talks about where this project can go in the short term and how the features could further be used to enhance learning. Lastly, Chapter 8 ties everything together and explains some lessons I have learned throughout this project.

Chapter 2

TaleBlazer

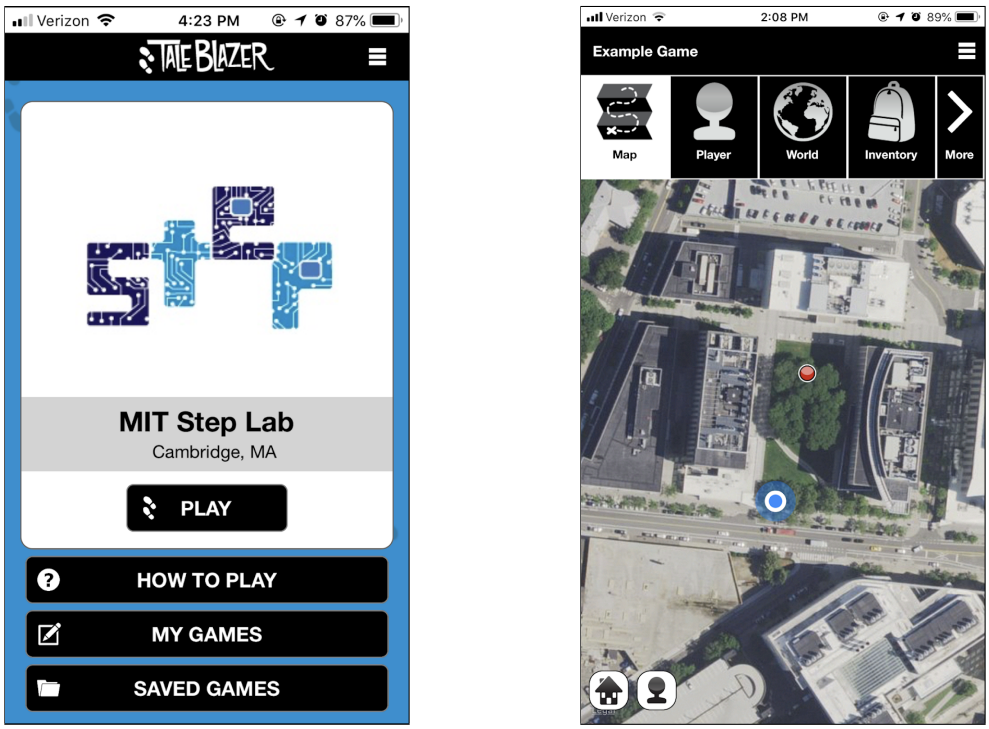
TaleBlazer is a location-based augmented reality platform for educational mobile games. In this chapter, I will first explain the different types of TaleBlazer users, then the game concepts that are especially relevant to my project, and lastly I'll walk through a game to better explain typical game mechanics.

2.1 Users

There are two main components to TaleBlazer's platform: the mobile app and the online web editor. This allows users to: 1) learn about the educational concepts wrapped into the gameplay experience as a player, and 2) learn about the computer science concepts behind making games as a game developer. These are two related but also distinct ways to interact with the platform; a user might take on only one or both of these roles.

2.1.1 Players

As a player, a user can download and play games in the TaleBlazer mobile app, which is freely available for iOS or Android. Through the app, a player can access featured games that are located nearby (these are usually example TaleBlazer games or featured games by a nearby partner organization) or games that they have created



(a) The homescreen for the mobile app.

(b) An example of a TaleBlazer game.

Figure 2-1: The homescreen of the TaleBlazer mobile app (left) and an example of a TaleBlazer game (right). In the right image, the blue icon represents the player locations, and the red icon represents an agent in the game.

on their account. A user can also access other games by using a game code, which is a unique sequence of letters. As shown in Figure 2-1a, when a player first opens the map, they will see the TaleBlazer homescreen for the mobile app. After a player finds a game and opens it, they might first see a map with their location represented by a blue circle, along with other icons, which represent in-game characters or objects called agents, as shown in Figure 2-1b. As a player walks around in the real world, they can trigger events with agents by approaching that agent's location on the map; this is called bumping the agent. When a player bumps an agent, that agent's dashboard will appear, which is a screen with more information about the agent and/or ways to interact with the agent (an example of an agent dashboard is shown in later in Figure 2-5).

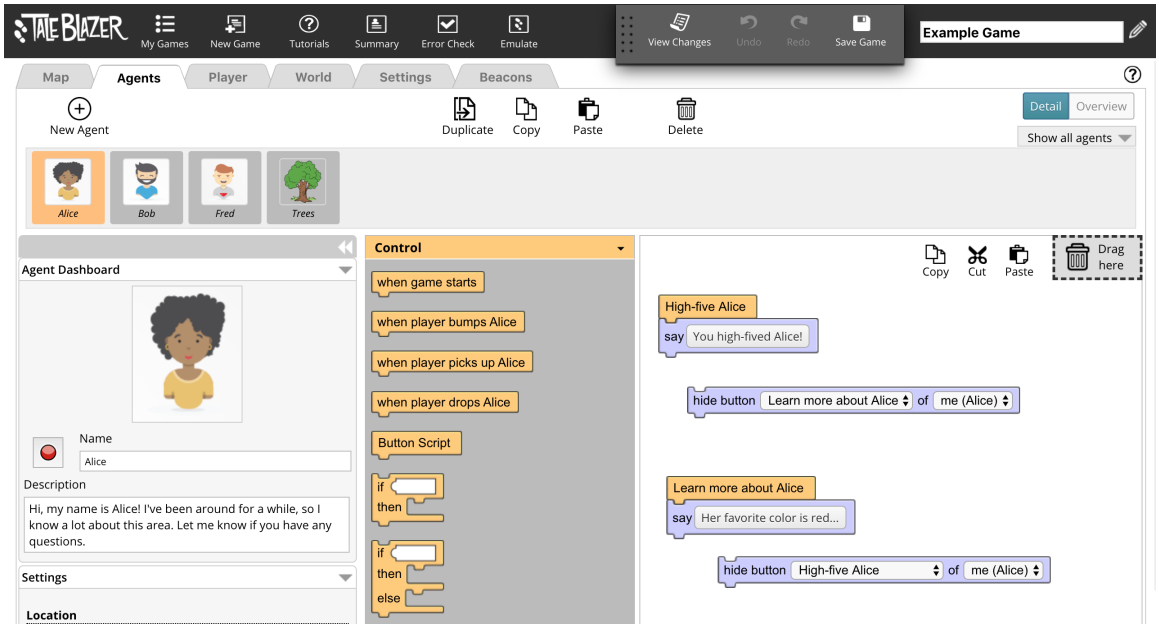
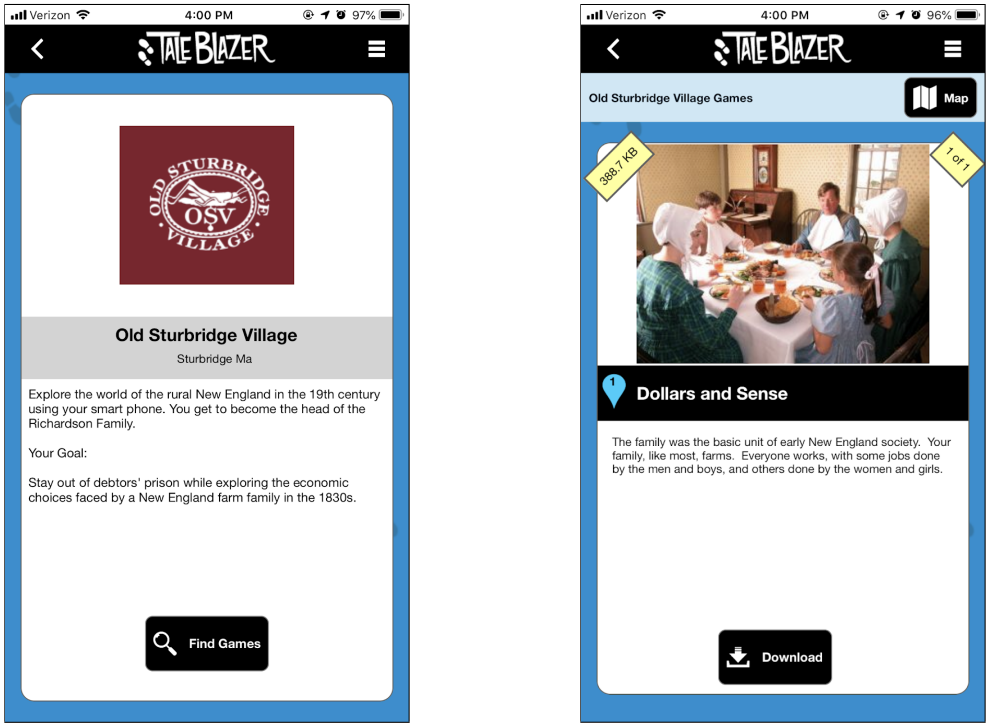


Figure 2-2: TaleBlazer's online web editor. The current tab is for the Agents view.

2.1.2 Game Developers

As a game developer, users create games using TaleBlazer's online game editor. As shown in Figure 2-2, the editor has multiple tabs (such as Map, Agent, etc) that open different views and allow game developers to easily customize the gameplay experience. The editor uses blocks-based programming (as shown on the right side of Figure 2-2) to let game developers add logic to their games, while also allowing them to learn programming concepts in a more user-friendly environment. Blocks can be pieced together visually like puzzle pieces and, rather than having to memorize syntax, users simply need to find the right block and move it into place. This is similar to other blocks-based programs such as Scratch or App Inventor. TaleBlazer's online game editor makes it easy for novice developers to customize a user's game experience, making it a useful tool for education in classrooms.

In addition to being used in classrooms, TaleBlazer partners with organizations to help develop engaging educational experiences for specific locations, such as historical sites or museums. These partner organizations can be considered a more specific type of game developer for TaleBlazer, because they are also involved in the development of a game for their specified locations. As a prime example, TaleBlazer features a



(a) The organization page.

(b) The organization's featured games.

Figure 2-3: Example of a partner organization page and their featured games on the mobile app.

game called "Dollars and Sense" which is located at the Old Sturbridge Village, New England's largest outdoor living history museum. Old Sturbridge Village is a partner organization to TaleBlazer (officially referred to as one of the TaleBlazer "Places"), and their game "Dollars and Sense" is one of their featured games. Using TaleBlazer, students can learn about the typical finances and decisions involved in the lives of old New England rural families by making decisions to buy and sell throughout the game. The game not only engages students beyond just passive listening, but it also immerses students to teach them about old New England lifestyles. Games created by partner organizations are generally featured and can be easily found if the game is located nearby. Figure 2-3 shows Old Sturbridge Village's page and games, as seen on the mobile app.

2.2 Game Concepts

There are many different features in the TaleBlazer platform, but I will focus on the ones that are most relevant to my project: the map, the agents, and the game tabs. For each of these features, I will first explain how players interact with the feature and how game developers can customize the feature. Then, I will explain what a typical TaleBlazer game looks like and how they are generally played to give a sense of typical game mechanics.

2.2.1 Maps

One of the main features of any TaleBlazer game is its map, which shows the location of the player and the location of other agents in the game; as a player moves around in the real world, the location of the player icon on the map gets updated. On the online editor side, a map can be customized in the Map view, which is shown in Figure 2-4. In the Map view, a game developer can drag agents around to place them somewhere on the map. They can also specify one or multiple regions of a game, which can each hold their own set of agents; this is useful if a game spans a large geographic area because a player only sees one region on the map at a time. Each region can be designated as a dynamic map or a static map. If a dynamic map is used, the game requires a data connection to load the map. If a static map is used, a data connection is not required because the game developer uploads a static image to the game instead.

2.2.2 Agents

As mentioned earlier, another key component of TaleBlazer games are the agents. When a player bumps an agent, they can see the agent's dashboard, which is shown in Figure 2-5. An agent's dashboard can have three sets of characteristics: information about itself (image, name, description), traits (attributes and their values), and actions (buttons the player can tap to initiate some action). These characteristics are labelled and shown in Figure 2-5. The information about itself includes the agent's

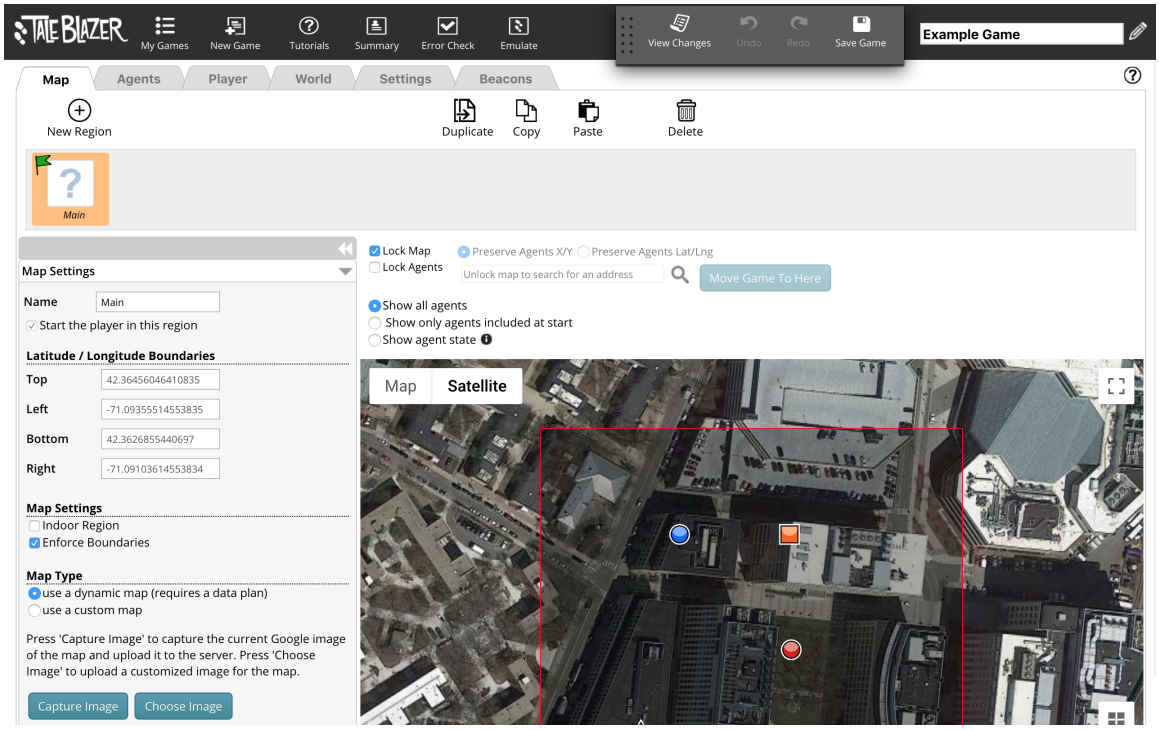


Figure 2-4: The Map view of TaleBlazer's online game editor. The option to use a dynamic vs. a static map can be seen under "Map Type".

image, the agent's name, and a textual description of the agent. Traits are attributes associated with an agent, and these attributes have values which can change as the game progresses; changes are specified in the blocks. Actions are large buttons that appear on the dashboard that allow a player to initiate some action with the agent. The logic for what happens when a player presses an action button can be defined in the blocks as well.

A game developer can customize agents and specify any logic associated with agents in the Agents view of the web editor, which is shown in Figure 2-2. New agents can be added, and their dashboard information can be customized on the left panels. That is also where game developers can add and customize the traits and/or actions of an agent. In the middle is the blocks drawer, where game developers can pull out blocks and place them to the area on the right, where blocks logic is defined.

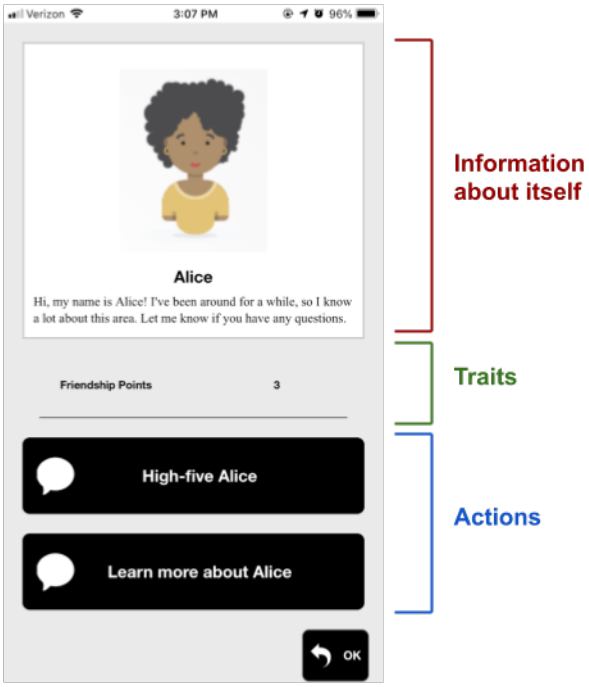


Figure 2-5: An example of an agent's dashboard.

2.2.3 Mobile Tabs

Another important aspect of TaleBlazer are the tabs that appear in the game on the mobile app. These mobile tabs are the main control of the game, and they allow players to switch back and forth between different screens that provide different information. The tabs that are most relevant to my project are the map, player, inventory, and history tabs.

- Map tab: On this tab, a player can see their location and the locations of agents in the game. This tab is generally used to guide players and show them where to go next.

- Player tab: On the player tab, a player can see information about their role and any traits that might be associated with themselves.

- Inventory tab: On the inventory tab, a player can see the list of agents that are in their inventory. These will usually be agents that represent objects or items in the game. A player can still interact with agents from the inventory.

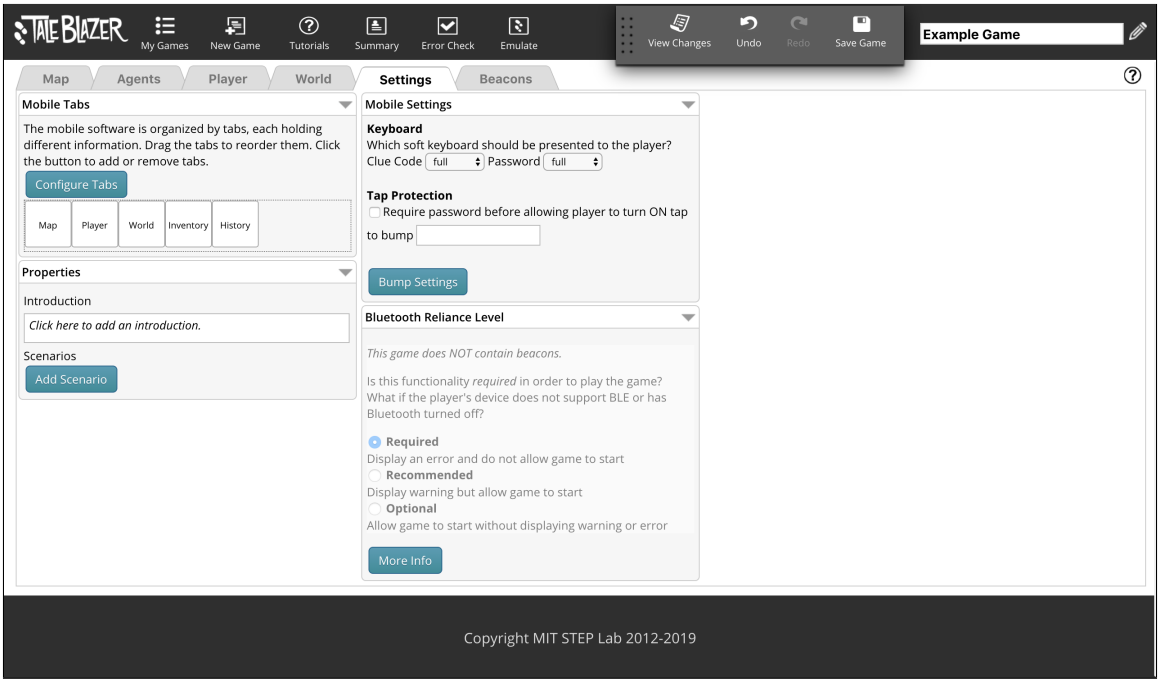


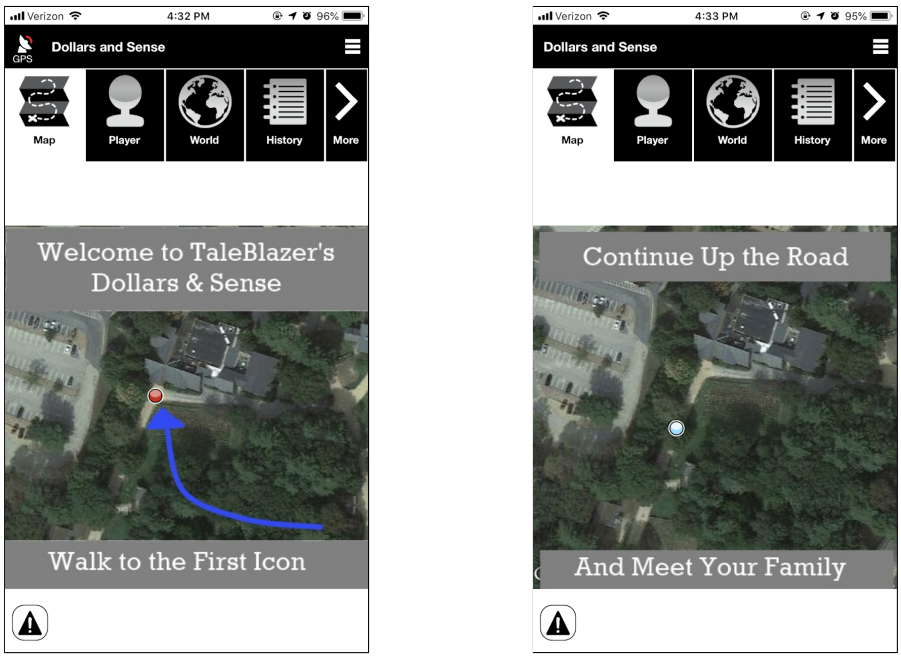
Figure 2-6: The Settings view in the online editor. The section to configure tabs can be seen in the "Mobile Tabs" section.

- History tab: On the history tab, a player can see the list of agents with which they have interacted in the game. A player can only see a snapshot of what they have seen most recently after visiting this agent, and they can not interact with the agent in a new way from the history tab.

A game developer can customize the tabs that appear in a game in the Settings view of the web editor, which can be seen in Figure 2-6. This is where game tabs can be added, removed, and reordered. Currently, there is a set list of possible tabs that can be used in the game. This view also allows game developers to customize global game settings, which are not specific to any particular object or feature in the game.

2.3 Typical Game Scenario

TaleBlazer games are typically storytelling games where players can interact with agents to progress the story. To help explain how a typical TaleBlazer game might work, I will walk through "Dollars and Sense". In this role-playing game, a player



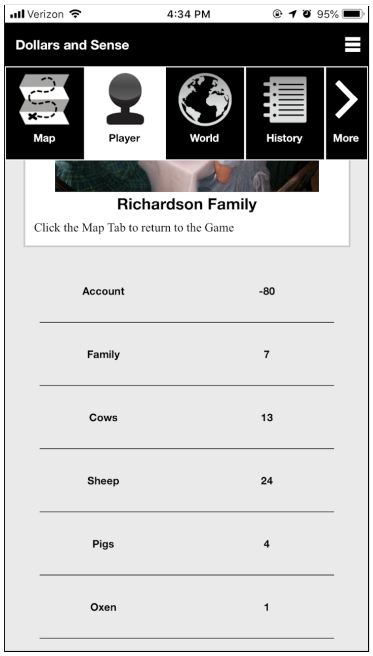
(a) The map tab before bumping the first agent. Only the first agent is shown on the map.

(b) The map tab after bumping the first agent. Only the second agent is shown on the map.

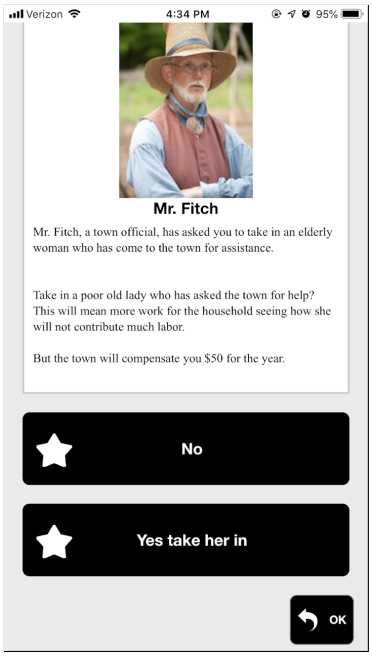
Figure 2-7: Example of how the map tab might update as a player progresses a game. The images show the map before (left) and after (right) a player bumps the first agent in the game.

gets to experience the lives of typical New England families, making decisions to affect the end outcome. When first opening the game, the player is taken to a static map of the area, with a single agent on the map. After a player bumps that agent, a new agent will appear on the map, which leads the player to walk to a new location as shown in Figure 2-7. When a player bumps an agent, they might also be required to make a decision by choosing one action out of several. Figure 2-8b shows an agent that requires the player to make a decision.

As the player progresses and makes decisions throughout the game, the player's attributes (e.g., bank account or number of members of their household) reflect changes from those decisions, as shown in Figure 2-8a; in this way, a player's choice can impact the game. The game will continue like this, with the player bumping agents and progressing the storyline until the player gets to the end of the game. The end of a game is typically signified by changing the map region to one that is designated as

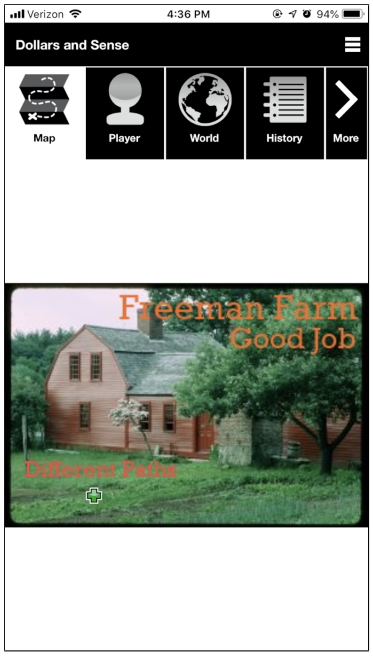


(a) The player tab in the game, which displays player information.

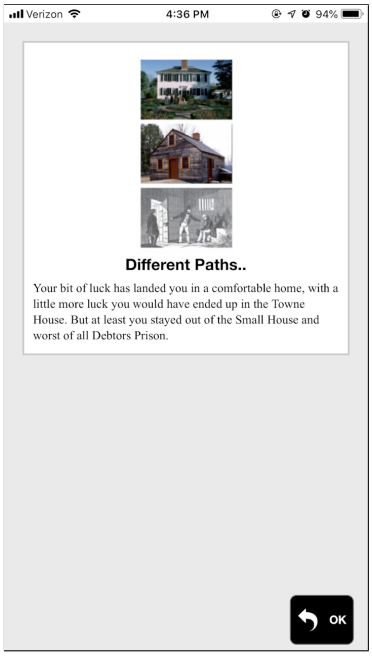


(b) An agent that prompts the player to make a decision.

Figure 2-8: Example of traits for a player on the player tab (left), and example of an agent that requires the player to make a choice (right).



(a) The final map of the game.



(b) The different outcomes of the game.

Figure 2-9: The final map region of "Dollars and Sense" (left), and the different outcomes players could have ended with (right).

the final map region by the game developer, an example of which is shown in Figure 2-9. Typically, a game developer can upload a static map with a custom image that is different from a picture of a map to help the player realize the game has finished.

This is by no means the extent of TaleBlazer's features, but it does demonstrate a common use case for TaleBlazer game mechanics which will be helpful for understanding my design considerations.

Chapter 3

Citizen Science

TaleBlazer aims to create engaging educational experiences for players and relies largely on storytelling to achieve that; citizen science also aims for engaging educational experiences through active learning about scientific research and observations. Citizen science is the involvement of the public in research, usually in the form of collecting and/or analyzing data [1]. This makes it possible for anybody, no matter age or occupation, to participate in scientific projects, make discoveries through collecting data and analyzing trends, and develop an interest in their surroundings and communities. In addition, the crowd-sourcing of data and information allows scientists to develop new hypotheses and make new observations through the help of hundreds and thousands of volunteers.

3.1 Projects and Impacts

There have been a number of significantly widespread citizen science projects, with some examples of earlier projects including the Lost LadyBug Project and the Monarch Larva Monitoring Project [2]. Both projects were designed to observe and track wild insects; the contribution of the general public allowed scientists to gather thousands of sightings and pictures that, if even possible, would have taken much longer to collect with the effort of only a handful of scientists. More recently, citizen science projects have also been moving towards mobile apps, such as the iNaturalist app and

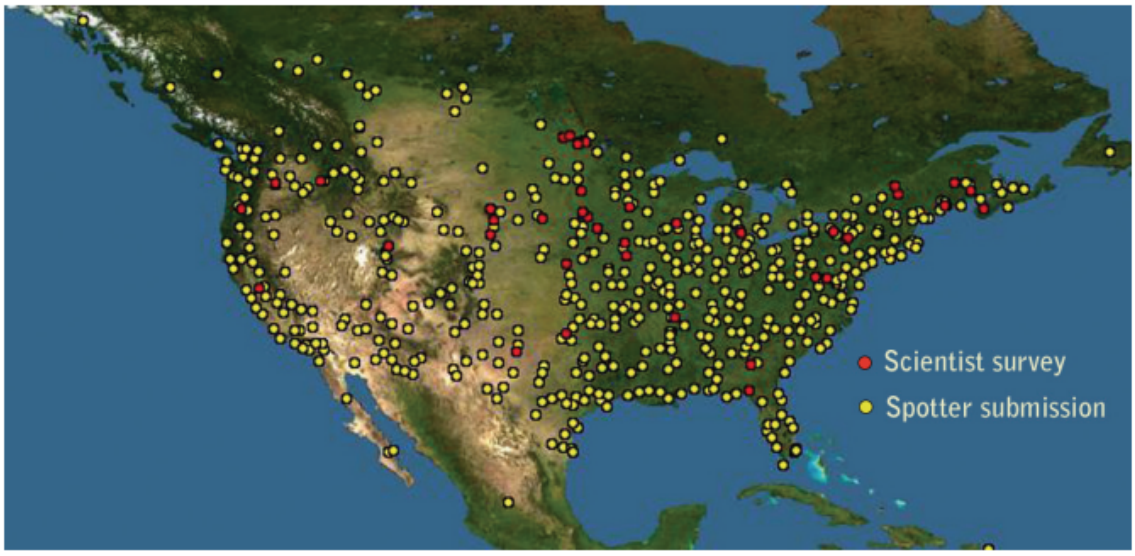


Figure 3-1: A figure from a published paper, showing the number of ladybeetle observations recorded in scientific surveys and spotter submissions.

the LeafSnap app. Both apps aim to collect information about nature, and they make it simple and easy for users to capture photos wherever they go. In addition, websites such as SciStarter (scistarter.com) and CitSci (citsci.org) allow anybody to start citizen science projects and invite wider communities to participate and contribute [5].

An example of the impact of citizen science in both research and the education of those participating can be found in the Lost Ladybug Project. Using the information obtained through the project, researchers were able to make interesting observations that they could not make before, with the collected information used in several publications [4]. Figure 3-1 is from one of those publications and explains how effective the citizen science project was; the contributions from citizen scientists (called "Spotter submissions") can be seen to cover a large geographical region. The study admitted that scientist surveys captured more data than typical spotter submissions but that spotter submissions covered more breadth, and that might have been what contributed most to bringing new information about ladybug locations and migrations [3].

Some key findings of educational impacts of the Lost Ladybug Project can be

found in a report from the researchers at Cornell who started the project [6]. Both children and adults involved showed an increased knowledge about ladybugs: adults showed an increased knowledge of ladybug biology and diversity, and children showed increased knowledge of ladybug vocabulary. The study also found that children displayed skills relating to data collection and understanding of the purpose of the study, while skills in data analysis and inference proved to be lacking in general. Additionally, the report found that a strong motivation for both children and adults was being able to make useful contributions to research.

3.2 Apps and Websites

More recently, citizen science projects have also been able to take a more flexible approach with apps and websites. The apps that most directly take inspiration from citizen science are environmental apps such as iNaturalist and LeafSnap. As briefly mentioned in the previous section, these apps allow users to better understand the natural world as citizen scientists. In the iNaturalist app, users are encouraged to observe animals and plants by taking photos and sharing them with a community of other nature-enthusiasts. Once an observation is shared, others can help identify what the animal or plant species is in the photo. These observations are also tagged on an interactive map, which allows users to see what types of species are common in their neighborhood (or places they visit) and find out where they might commonly see the species. In addition, because these photos are tagged with the species name, location, and time the photo was taken, these make for very useful data points for scientists who may be interested in a species or even creating a dataset of photos of certain species. In the LeafSnap app, users can take photos of leaves and have the app suggest some types of leaves that match the leaf's appearance in the photo. In addition to helping users learn by assisting in the identification of local plants and trees, the app provides some games to help train the users on identification as well. This is something that is more typically seen when researchers try to crowdsource help for activities such as photo identification - they need to equip the volunteers

with the skills to know how to identify and label such pictures.

This is especially common in other citizen science projects that have been gamified. Projects such as Citizen Sort and Project Nightjar are a couple examples. A user can go to the website for Citizen Sort and begin participating by identifying animals in a photo. Users try to correctly identify animal species in photos that have been taken by cameras with motion detection sensors. For the game aspect, users are given a set number of lives and a score for how many they succeed to identify. There is also AI that has already classified the animals in the photos, and if a user thinks the AI misidentified an animal, they can mark the photo with the correct identification along with a comment about why it is incorrect. This helps the researchers, designers, and developers behind Citizen Sort, a team from Syracuse University's School of Information Studies, confirm how well their AI seems to be working. For Project Nightjar, scientists both create awareness about nightjars, which are a species of birds, and give users an opportunity to check their speed in being able to identify the birds who are camouflaged into the background. This also helps researchers and scientists understand how well predators may be able to perceive these species.

3.3 Interview with an iNaturalist User

In trying to understand the motivations and incentives behind participating in citizen science activities, as well as understanding the effects of such activities on an individual, my team found and interviewed an enthusiastic user of the iNaturalist app. We found several interesting points from this interview, and I will highlight several here.

In summary, we found the reason the interviewee liked using the app had a lot to do with self-discovery and improvement. They felt motivated by the fact that their pictures could be used for research purposes and that they were able to look back and identify different trends from the various observations that they had made. In addition, they felt that the activity gave them the time and space to observe - to see and hear all the things around them, which was an activity that they hadn't engaged with before. It gave them a bigger appreciation of even the different species in their

backyard.

This further motivates our project and helps us understand what aspects of citizen science might be worth including. The interviewee really enjoyed aspects of iNaturalist that seemed more like a game or an assessment of skill. For example, the interviewee felt motivated by the "grading" of the photos and wanted to perform better to get better "grades". They also stated they would have enjoyed something in the app that would have encouraged him/her to aim for something, which is similar to what one might consider quests or scores in games. Most importantly, the interviewee enjoyed the aspect of being able to make discoveries and identify trends by themselves. This means it might be helpful to include some way for players to analyze the things that they have observed while playing a game.

Chapter 4

Motivation for TaleBlazer

There are several motivations for this project, beneficial both for TaleBlazer users and its partners. One primary motivation for this project is to allow users to learn more about science and research using the TaleBlazer platform, and in doing so, connect with the world around them in meaningful ways. Another related but distinct motivation is opening the door for TaleBlazer games to be built upon player-generated content instead of solely relying on game developer-generated content. TaleBlazer games currently place the burden on the game designer to define, a priori, all the elements of the game, including the real-world locations of all the elements in the game.

4.1 Citizen Science Learning

Currently, TaleBlazer allows for games that are very narrative-focused, such as games teaching history or cause-and-effect of events, but it is very limiting towards games that allow players to record meaningful information about the world around them. As mentioned in Chapter 3.1, researchers with the Lost Ladybug Project found positive impacts of the project in the knowledge of ladybugs that participants had. Because TaleBlazer is a generic game-making platform, rather than focusing on a specific topic, I focused on adding functionality that could enhance a student's abilities to understand the process of research, including both data collection and data analysis.

The educational impacts of a game would vary depending on each individual game, but the addition of these features would open the doors for creating games with these goals in mind. An important factor that was mentioned both in the results of the Lost Ladybug Project and our interview with an iNaturalist user is the notion of self-discovery and feeling like one is making a contribution. Currently, there is no way for a player to make unique discoveries in TaleBlazer; everything a player sees or interacts with in the gameplay experience is predefined by the game developer.

4.2 Portability

Because gamedevelopers must predefine all game content, it is very difficult to take one TaleBlazer game, which is designed for a specific location, and adapt it to another location. TaleBlazer games are intended to enhance a player's real-world experience, so games are conceptually tied to a certain location; when trying to move the physical location of a game, a game developer not only has to explicitly link together all of the individual agents of the game with new real-world locations, but they must also ensure the game still makes sense. For example, if an agent referenced a building in the area, the game developer would have to make sure that there was still a building close to that agent to reference. However, if players had more ways to input information, games could then use that player-generated content to shape the game rather than relying solely on static information determined from the start by the game developer. For example, if a game needed an agent to be placed next to a tree, the player could identify and specify the location of a tree in the real world, and the game could use that information to place the agent, rather than requiring a game developer to know exactly where a tree was for the location of every player who wanted to play.

Using the features developed during this project, the TaleBlazer platform will have functionality readily available for game developers to create citizen science project-like games. Players will be able to collect and analyze information during gameplay. In addition, players will be able to incorporate information about the world around them into the game and potentially make it easier to port games from one location

to the next, by not leaving all the burden on the game developer.

Chapter 5

Goals

In the designing of the new features, there were many trade-offs I had to make between different designs and implementations. There was not always an obviously correct or better way to handle something, so I prioritized the most important goals to TaleBlazer and to this project. In this chapter, I'll describe the targeted use cases my team and I aimed for and highlight some design goals we had in mind as well.

5.1 Targeted Use Cases

In considering different use cases and different possible games, we narrowed the use cases down to two different types of games:

1. Data Collection Game: This might be a game that is very similar to iNaturalist, allowing players to decide when and where they would like to enter data and be able to review this data. This type of game would be the most "citizen science"-like use case.

2. Sequential Decision Making: This might be a game that is similar to current typical TaleBlazer games. A player can input information to make choices throughout the game, and these choices would directly impact gameplay. This type of game would be more focused on blending form information into the "gaminess" of TaleBlazer.

In the end, there were two main types of games we wanted to support - a data collection game and a sequential decision making game. We found that these were the two distinct use cases of having players inputting information into the game. This also factored into my decision to create two different types of agents with different purposes because there are three distinct differences between how players would interact with a form in these two cases.

5.1.1 Interaction Initiation

The first difference is in how the interaction would be initiated. Typically, an interaction with an agent can be initiated in one of several ways. One way is through approaching an agent's location in the real world, another way is by the game developer running a script that has the player bumping the agent after certain events, and yet another way is by tapping on the agent icon on the map (this method is typically used for debugging purposes for the game developer). In a data collection game, a player should be able to control when they input information because they are the ones who decide if and when they see something of interest; a player should be able to input information anytime and anywhere. On the other hand, in a sequential decision making game, a player would not have to input information unless they are prompted to make a decision and ask a question by an agent.

5.1.2 Number of Submissions

The second difference, assuming the player can fill out a form to input information, is in how many times a player might submit the form. In a data collection game, a player might make multiple observations of the same type, where each type of observation has its own form to submit. In this case, a player should be allowed to fill out and submit a form multiple times. In a sequential decision making game, a player might only have to make a decision or answer a question once. It would not make sense for the player to go back and make another decision or answer the question in another way because the game would likely be following a narrative.

5.1.3 Reviewing Submissions

Lastly, there are differences between how the player might want to interact with the inputted information. In a data collection game, a player might like to see the collected information in some meaningful way; this might include seeing the submissions marked on the map where they found the point of interest or seeing them all in a list. In a sequential decision making game, players would typically be able to see the impact of their decisions directly in gameplay, and at most, a player would need to be able to see the history of decisions they have made but not in a collective view.

5.2 Design Considerations

Throughout the design process, my team and I prioritized the following three goals when making decisions. First, we wanted to make sure that TaleBlazer remained a user-friendly and flexible platform. Second, we wanted to emphasize simplicity of design and usability. Lastly, we wanted to remain focused on prioritizing both the education value and the fun value of the TaleBlazer platform.

5.2.1 User Friendliness and Flexibility

TaleBlazer users can range in age from elementary students to adults, and their expertise level can vary from beginner to expert. We wanted to ensure the platform remained flexible enough to be easily used for beginner users and well-equipped enough for advanced users, for both players and game developers. This meant a lot of decisions were made to ensure the most generic use case was obvious and easy, while hiding away some of the more complex features. These features would ideally still be easily found in the tutorials and in sample games for expert users.

5.2.2 Simplicity

We also wanted to emphasize simplicity of design and usability. TaleBlazer has a rather large codebase and has had many different software developers working on the

code, and there will likely continue to be more working on the code in the future. For this reason, its design must be easy to understand and ready for change. In addition, in accordance with our goal of user-friendliness, the usability of TaleBlazer should remain simple. As long as something works and meets 90% of our possible use cases, there's less importance in getting that last 10% if it adds too much complexity to both the codebase and the feature's use.

5.2.3 Playful Learning

Lastly, we wanted to prioritize both education and playful engagement for this project. While it was important to add features that would expand the potential for education, it was also important that users could still feel like they were playing a game and not just filling out information in the midst of gameplay. We wanted the new features to enhance the learning experience in gameplay while not taking away from the playful and engaging experience of it all.

Chapter 6

New Features

There are three related but distinct features I designed and implemented to incorporate concepts central to citizen science into TaleBlazer games. First, to allow game developers to ask players for specific information, I implemented a custom form and added it as a characteristic to agent dashboards. Second, to represent the two distinct use cases for player input described in Section 5.1, I designed two different types of agents that can hold the form. Lastly, to display the information collected through these forms, I added and modified the tabs in the mobile app to display collected information in a meaningful way.

6.1 Custom Form

Something that became evident after looking at various citizen science apps and websites was the need for some way for players to input information. However, one of the biggest weaknesses towards scientific learning in TaleBlazer's current platform was a lack of a way for players to do just that. On an agent's dashboard, there are predefined actions (as described in Section 2.2.2) that can reflect a player's choice. There are also password-protected agents and a cluecode tab, which require a player to input text that must match a predefined keyword. However, there is no way for a player's custom inputted information to be saved and used in a game. In addition, because TaleBlazer is a platform for making all kinds of games with different themes

and purposes, there's no set piece of information that would be useful across the board; it is arguably necessary to give game developers the flexibility to decide what information is relevant to their game. To address these issues, it became evident that there was a need for some type of custom form, where game developers could customize what type of information a player could record.

6.1.1 Form Creation Software

To get a sense of what might make intuitive sense to users, I took a look at some existing form creation software - Google Forms and Survey Monkey. Particularly, I focused on the types of questions they offered, as well as the user interface and experience (which is explained later in Section 6.1.2).

Types of Input

In the process of creating a custom form creation software, I considered different types of inputs and tried to include only those that would make sense in TaleBlazer's platform. I first looked at what types of inputs were common to both Google Forms and Survey Monkey; this gave me a good breadth of options to think about because both platforms cater to many different uses. Some of the types of questions that were common to both platforms were: short text, long text, multiple choice/radio, dropdown, checkboxes, date/time, and file upload. From this list, I refined the options even further by thinking about what issues were faced by TaleBlazer users and what information would be helpful to collect.

In a data collection game, a player might want to write notes about or take photos of what they have observed. For this reason, I decided to include short text, long text, and photo input types. Both the short text and long text input types were common to both the Google Forms and Survey Monkey platforms. Photo input was not common to both platforms, but a file upload type was; in TaleBlazer's use cases, players would have no need to upload generic files, but photos would indeed be helpful.

Although not implemented for this project, it is possible that information from

these forms might get aggregated and used for analysis in the future. To make it easier to analyze inputted information, it would be helpful to have input types that were more limiting, as opposed to text and photo inputs which are free-form and could be anything. For this reason, I have included number, radio/multiple choice, and checkbox input types. The number input type would appear the same as a short text input type, but the inputted information would be limited to being a number; this would be useful for aggregating information into, for example, sums, averages, minimums, or maximums. The radio/multiple choice and checkbox input types would contain options that were predefined by the game developer. These input types would also be helpful for the sequential decision making game, where players might be expected to choose one or multiple options out of many.

Lastly, date and time input types seemed to be useful, especially if players are allowed to submit a form multiple times; however, I decided to make that a necessary metadata component of any submission of a form instead of an input field for the player. Google Forms and Survey Monkey serve all types of forms including RSVP forms and date planning forms which is why those fields might be presented to a user. However, TaleBlazer games would not need to gather that type of information from users. Instead we would want to use that information as metadata associated with a form submission.

Overall, I made decisions to include types of input that would help maximize simplicity and understanding for our users and game developers, while additionally considering our goal of enhancing scientific capabilities. The implemented input types are listed in Table 6.1.

Input Format Design

The next question I had to consider was how to present a form to the player. In alignment with our goal of wanting to keep the fun aspect of TaleBlazer, I did not want the player to feel like they were leaving the game to fill out a form. Instead, I wanted to give agents in the game the ability to "ask" the player questions, to create the experience of the player responding to an agent in the game, which is

Type	Options
Short Text	None
Long Text	None
Photo	Limit to 1, No Limit, Custom Limit
Number	None
Radio (Multiple Choice)	Add Option
Checkbox	Add Option

Table 6.1: A table of the available types of questions and their options (if applicable) in the custom form.

why we incorporated the form into the agent's dashboard. Typically, an agent has three sets of characteristics (as described in Section 2.2.2): information about self, traits, and actions. I considered adding inputs as an option to one of these existing characteristics but realized it would be rather restricting; the only suitable place for inputs would be in the actions, and having an input on top of a button would lead to a confusing user interface. In the end, adding capabilities for a new characteristic of custom forms seemed to be the best option.

In designing this feature, there were two distinct components; it needed to both be supported on the online web editor for our game developers and the mobile app for our players. In the following sections, I'll talk first about how the game developer can create the form, followed by how the player can then interact with the new form and input information.

Implementation Decisions

Another question I had to consider was how to represent a custom form in the code. TaleBlazer's current representation of agents in the game is in a single Agent class, which extends the Entity class. Each Entity object is given a unique id and name, which is helpful in identifying the objects. I considered two options of implementing the form on the agent: the first was creating a different class of an agent which only contains the form, and the second was simply adding a flag (e.g., some field or attribute) to the existing Agent class that would indicate whether this agent has a form. I could have also considered having a child class of an agent, but difficulties

with type checking would have complicated the code; there were many places where an object's type of class was checked to perform certain operations, and I would have needed to include checking for the parent as well in these places. The problem with creating a completely different class of agent was whether or not this new class of agent also had traits and actions available. If this new class of agent were to also have traits and actions, it would lead to replicated code that would have to be maintained whenever one was changed (because we would want the traits and actions to be the same across both types of agents). However, if an agent were to have traits, actions, and a custom form, there would be a lot on the screen for novice users to process and understand. In order to keep a low threshold for using TaleBlazer, I decided, for my project, that an agent who has a form would not have traits and actions. On the other hand, it is completely possible that in the future there is a need for an agent to have both a form and traits. For this reason and also to maintain simplicity of the codebase, I decided to simply create a flag in the existing Agent class to indicate whether an agent had a form or not. The traits and actions of these agents are currently suppressed, but it would not be too difficult to undo.

I also had to consider how to represent each question on the form in the code. In following the examples of how traits and actions are each defined in their own Trait and Action classes, respectively, I created a FormElement class to represent each question in the form. In addition, I let this class also extend the Entity class, so each FormElement could have its own unique id, which is useful when elements are reordered from dragging and dropping on the editor. By following existing examples and extending existing classes, I hoped to make the code simpler to understand for software developers who may work with the same code later on.

In addition to creating new classes for new objects, I was met with various challenges in how to incorporate my code into the existing codebase. For example, for adding new elements into the game, there already existed a pipeline for adding Entity objects. This pipeline was rather convoluted and had many steps that went through different files, some of which were not necessary for adding a new FormElement object. However, if I were to create my own pipeline for specifically adding FormElement ob-

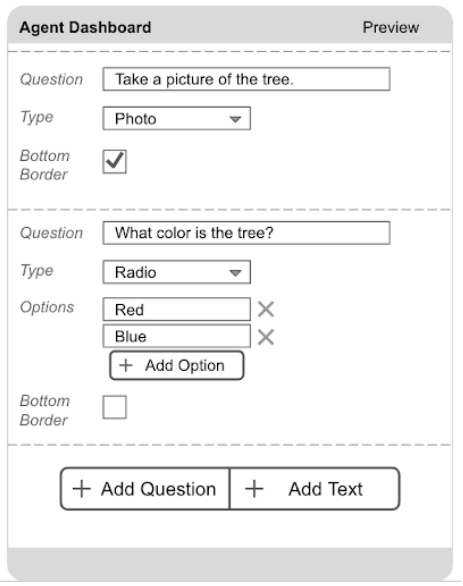
jects, there would be a new and separate entry point for adding Entity objects into the game that someone following me would have to know existed. This would complicate the codebase and make it more difficult for others to understand. After considering how my code might be understood by future software developers, I chose to follow the existing pipeline for my new FormElement objects. More generally, I tried to either mirror an existing function's structure or incorporate my code into existing pipelines whenever I could, for simplicity and ease of understanding.

6.1.2 Custom Form on the Editor

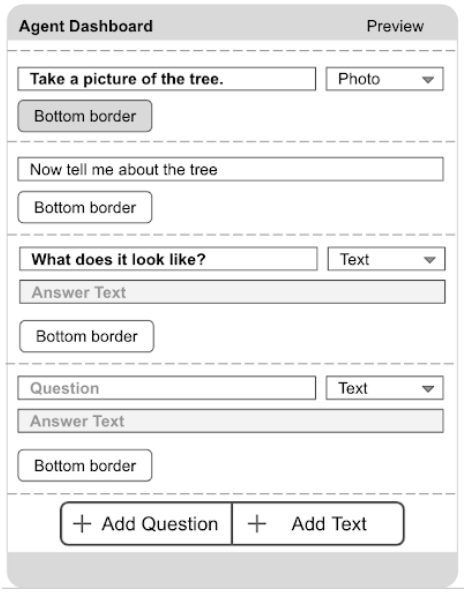
Allowing players to input information during gameplay starts from enabling the game developer to decide when and how they want players to do so. As mentioned previously, to make the custom form editor as intuitive as possible, I looked at Google Forms and Survey Monkey. Both interfaces had similarities in allowing a user to add a question with the click of a single button. I wanted to reflect this design to prioritize simplicity and user-friendliness for the game developer.

A difference between the form design of Google Forms and Survey Monkey was in the user interface of how to customize each question. Survey Monkey was more explicit in its design and had clear instructions for each field. Google Forms was more consistent in how the questions end up looking to those filling out the form. On the one hand, having less helping text, such as labels for each textbox, means there's less information on the page about what each box is for, but on the other hand, it would look more similar to the final layout on the mobile device. With these differences in mind, I mocked up two different versions of the form editor, as shown in Figure 6-1. Although the difference was subtle, I opted for having a more informative design to keep the floor low for our users, as shown in Figure 6-1a.

I also wanted to make the editing of the forms as intuitive and simple as possible. TaleBlazer already has some existing features that allow dragging and dropping in the movable Save Bar and the drag-and-drop Action buttons. To match the intuitivity of these features, I created a prototype showing how the elements could be dragged, as shown in Figure 6-2.



(a) Survey Monkey inspired mockup.



(b) Google Forms inspired mockup.

Figure 6-1: Two mockups showing the difference in possible UI on the form editor. The image on the left has more assistive text to help the user while the image on the right more reflects the final look of the designed form.

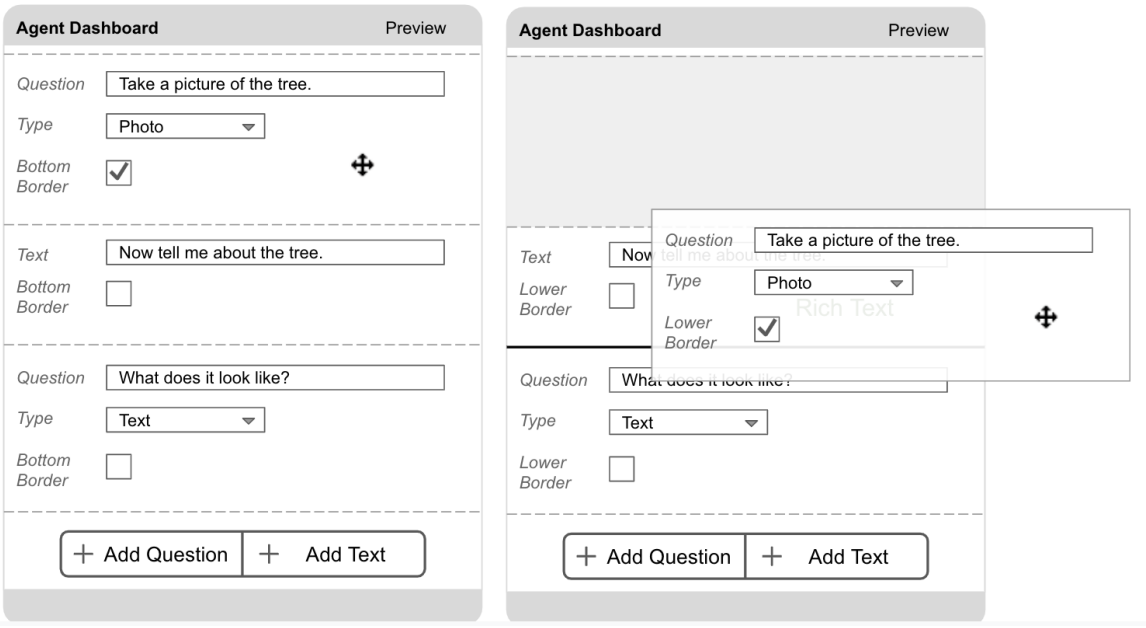


Figure 6-2: A prototype of how it might look to interact with each element of the form.

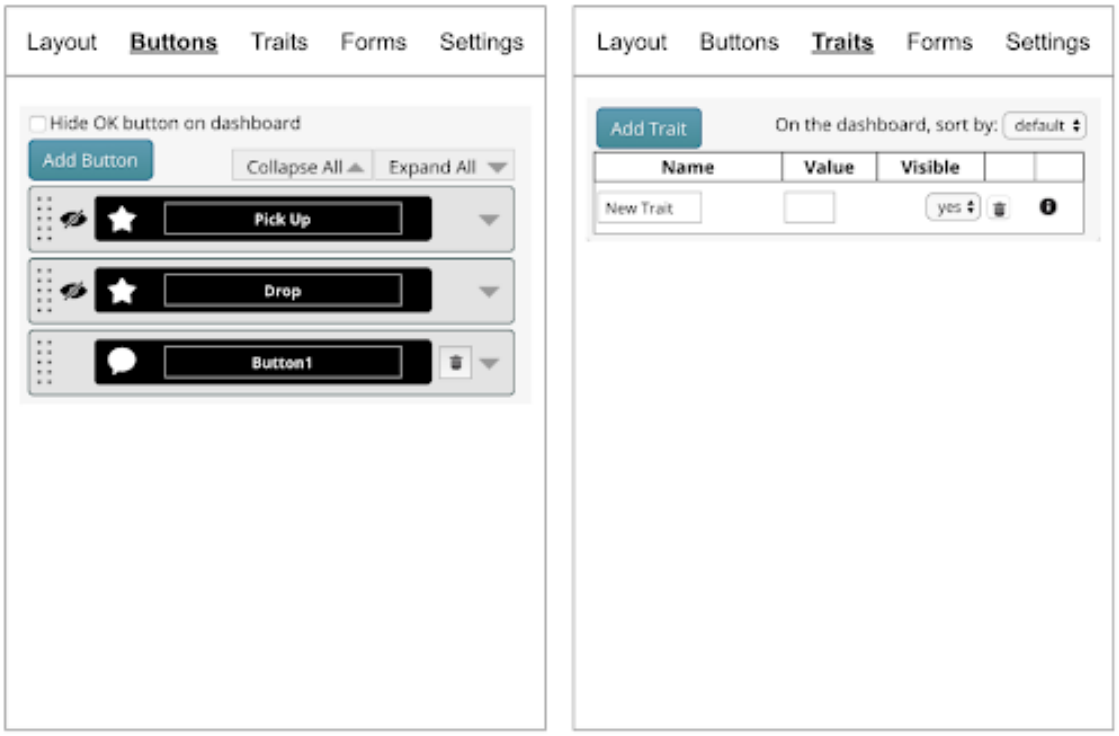


Figure 6-3: A prototype of how a tabbed form editor interface might look.

Adding this characteristic to agents also means adding more to the agent editor screen, so I considered some ideas on how to declutter the screen and simplify the editing experience for the game developer. One thought was to tab the characteristics so you only see one at a time, as shown in Figure 6-3. However, this might lead to confusion with tabbing within tabbing because the different views are already tabbed. Instead, we later explored creating different types of agents with different characteristics, as explained later in Section 6.2.

After an initial prototyping phase, I implemented the custom form in the online web editor. Because it was added as an additional characteristic for agents, it was simple in terms of implementation to follow similar patterns from actions and traits. To simplify the experience for the game developer, I've implemented a one-step add process, where the game developer simply adds a question or text section before customizing each section.

Each question or text section that gets added to the form is called a FormEle-

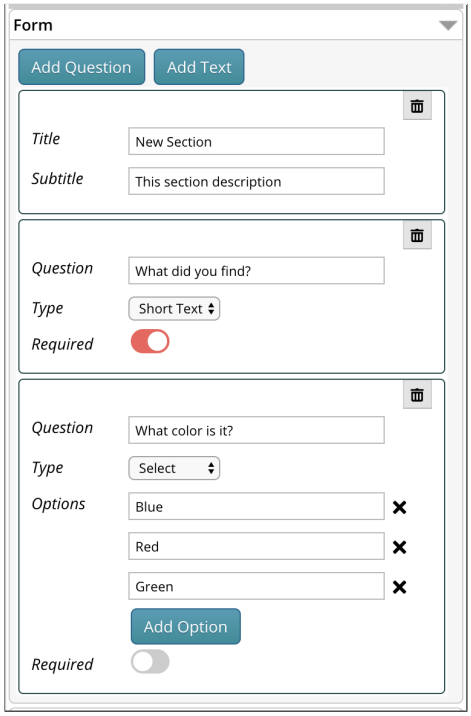


Figure 6-4: The implemented custom form on TaleBlazer's online web editor.

ment. Each FormElement can be dragged or dropped, and reordered by doing so. Each FormElement can be made required or optional with a simple toggle button, and a required question is indicated to the player through a red asterisk on the mobile app (as shown in Figure 6-5). This would be especially relevant in a data collection game, where there might be certain pieces of information that are necessary to an observation. There might also be some questions that could help the player record more details but which might not be necessary for analyzing all the information collected, which the game developer might want to leave optional. In addition, the TaleBlazer editor employs its own undo/redo operations, so I implemented the undo/redo functionality to maintain consistency with other features on the editor. The implemented custom form on the web editor can be seen in Figure 6-4.

6.1.3 Custom Form on the Mobile App

After providing a way for game developers to add forms to agents via the online web editor, I looked towards adding the relevant functionality to the mobile app. After a

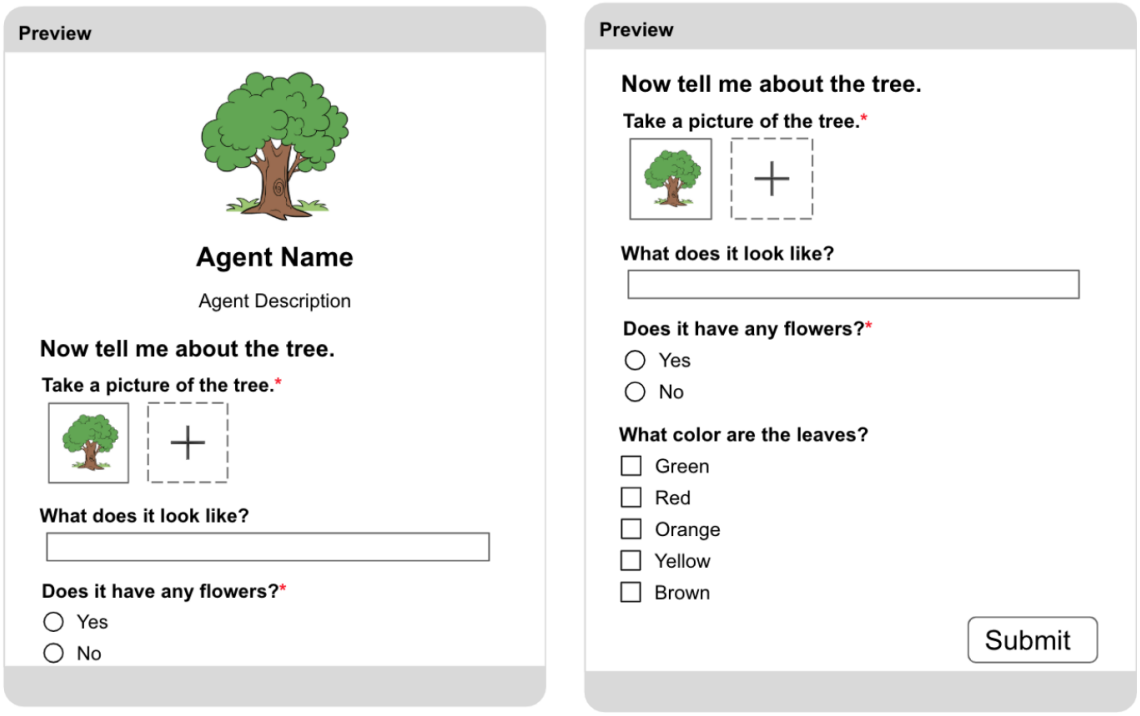


Figure 6-5: A mockup of the form on the mobile app.

player bumps an agent containing a form, they should see a form that they can fill out; this will be the form that was customized by the game developer on the web editor. A mockup of the mobile view can be seen in Figure 6-5.

Initially, I only had one type of agent in mind when I created this form in the agent dashboard, but my team and I eventually decided that there are two distinct use cases, as discussed later in Section 6.2. The main reason for this distinction came about when considering how the player would interact with the agent in the two use cases discussed in Section 5.1. In a data collection game, a player might want to fill out information for a form anytime they see something interesting. In this scenario, a player has control over when they see the agent and they would be able to submit the same form multiple times. Additionally, after submitting information, it would be helpful to see that information displayed or listed in aggregate. Alternatively, in a sequential decision making game, it may make more sense for a player to be prompted to fill out information during the game and to only be allowed to submit the form once. It would also be more interesting and useful if game developers could use the player-

inputted information throughout the game using blocks. Although both use cases were similar in asking the player for more information, there were differences in how the player would interact with the agent and the results of interacting with the agent. To make these subtle distinctions clearer, we decided to create two subcategories of agents who had forms.

6.1.4 User Testing

Throughout the design process, I was getting feedback from other members of the TaleBlazer team and my supervisors. However, we were all familiar with TaleBlazer, so it was important to also get another perspective on the new features. There were a couple instances throughout the year where I had a chance to get feedback from people who had never used TaleBlazer before. The first was another graduate student, and the second was a group of high school students who came to visit the lab. Both offered helpful perspectives and feedback for the online custom form editor. I ran both tests with a think-aloud protocol, where I had the participant(s) speak out loud as much of their thoughts and actions as possible. The introduction and task list used for the user tests is included in Appendix A.

Graduate Student

After completing the custom form feature, I had a chance to get feedback from a fellow computer science graduate student who was familiar with the design process. The student was tasked with creating a new agent with a form and was asked to add various questions with specific options. Through the testing, I wanted to see how intuitive and user-friendly the form was, both of which are very qualitative measures. Throughout the test, they were able to complete each task quickly and efficiently. Other than some cosmetic issues, such as difficulty dragging-and-dropping larger formElements or having to click twice to move the focus off an element, they seemed to finish the tasks with ease.

One more significant point for confusion was in their understanding of the differ-

ent options. They weren't quite sure what all the different options were and thought it might be helpful to have illustrations to help the user understand what they were selecting. Overall, the student claimed that it was pretty intuitive and said it "behaved similarly to Google Forms, which [they] were used to". However, they did wish it was more colorful because the interface felt very sterile although it was for creating games.

High School Students

After the first user test, I had an opportunity to receive feedback from high school students about the platform and current features. One change from the previous test was that I had a chance to create a mock up of what the form would look like on the mobile app, to test whether it matched their expectations. I tested with two pairs of people, for a total of four people. My team and I thought having a pair of people during each session would help the participants feel more comfortable both with the tasks and with thinking out loud by talking to each other.

One big mismatch of expectation was when I asked them to have the agent ask the player a question. Immediately, both groups went towards the blocks section of the editor. There were two factors that probably played into this expectation. One factor was that both groups had experience with blocks-based programs such as Scratch and App Inventor, and they explained that they "thought of the blocks as the brains". They expected to be able to define some logic such as, "if player bumps bob, then ask for his name". The other factor was probably not being able to see the form section immediately when creating a new agent. The form section can only be reached by scrolling down past the agent description section. I did not end up addressing this issue because it could be mostly alleviated with a tutorial about the form, but it may be something worth considering in the future.

After pointing them towards the form section on the editor, both pairs had an intuitive grasp of how to add new questions, how to change the questions, add options, set questions as required, and how to reorder the questions. There was a smaller confusion where they didn't quite understand what the question "type" meant. It

was not immediately clear to them that this meant the type of response the player could input. At the time, I also had the multiple choice question type called a "radio" type, reflective of the more technical name for having a multiple choice question. The participants understandably did not know what "radio" meant, so I changed the question type to be "multiple choice", which is definitely a more user-friendly term.

At the end of all the tasks, I showed them the mock up of how the form would look on the mobile app. Both groups felt it was rather similar to what they were expecting. However, both groups did claim they expected the questions to appear one at a time with a "next" button to go to the next question instead of being able to see all the questions at once. This was an option that I had considered with the TaleBlazer team for how to display questions to the user - in a "wizard" format. However, we strayed away from this design because it slows down the question answering process when players have to press buttons to go between questions when the questions don't rely on each other.

The participants came in with a little to no understanding of TaleBlazer; they were given a chance to briefly play through a sample game, along with a brief introduction to TaleBlazer concepts as a background introduction. In practice, a TaleBlazer user could learn about the functionality in more detail through the tutorials on the side and they might also be more familiar with the agent interface. However, the interview was still very helpful because I wanted to see how intuitive the interface was to new users and what their mental picture of the outcome might be.

6.2 Different Agent Types

As mentioned in previous sections, agents were divided into different types based on their purpose in the game. This helped both with decluttering the Agent view for the game developer and for clarifying an agent's purpose and associated actions for any TaleBlazer user.

As we continued working and talking about end use cases, we defined two subtle but distinct ways we wanted players to interact with agents. The first case is for the

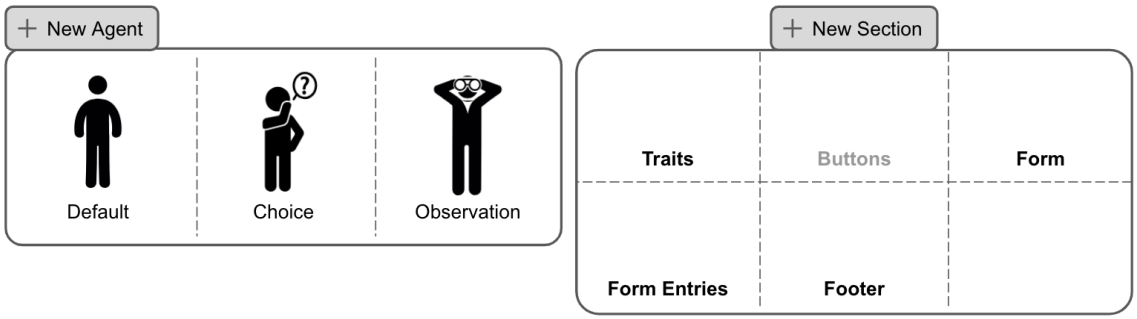


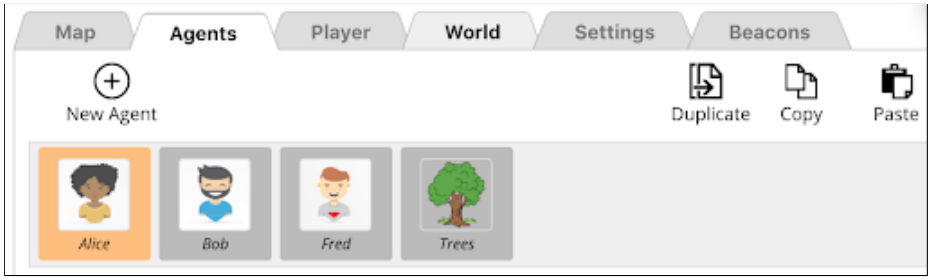
Figure 6-6: A mockup of having templated agents with customizable characteristics (called a "Section" in the mockup).

end use case of a data collection game where players would be able to make multiple submissions of the same form and possibly review all the information they've collected - we call the agent that handles this the Tag Agent. The second case is for the end use case of a sequential decision making game where players might want to visit an agent and submit their answer to a question once - we call the agent that handles this the Choice Agent.

6.2.1 Agent Types on the Editor

I originally considered having something similar to templates when creating new agents, similar to Google Slides and Microsoft Powerpoint. Game developers could quickly select a default agent template, but they would also be given the option to add and remove different agent characteristics, as illustrated in Figure 6-6. However, there would be added complexity when trying to add and remove different characteristics, so I decided to simply break them up into separate agent types without a way to move different characteristics between different agents. I chose to prioritize simplicity in design and implementation in this case. In the future, if someone wanted to add more flexibility for the game developers, it would not be too complicated to extend the feature such that each agent could have custom characteristics; however, it would require more affordability for the game developers and a more intuitive interface.

There were several designs that I considered for how to add these new types of agents, but ultimately, I kept to the same design as the editor currently has and added



(a) The existing "Add Agent" button.

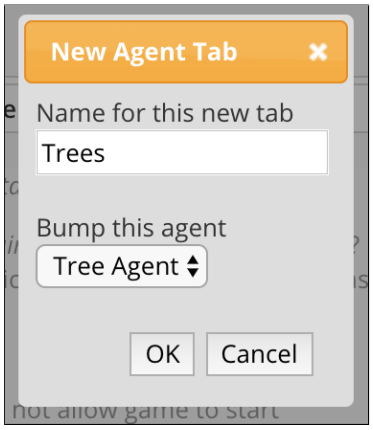
(b) The new buttons for adding different agents.

Figure 6-7: Screenshots of how the game developer could add agents before (top) and after (top) implementing different agent types.

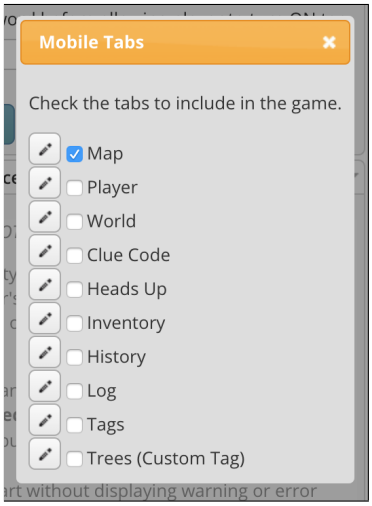
buttons in addition to the existing one. This was to make it easier for both new and current users when creating these agents. Current users are used to the one "New Agent" button. By keeping that button there, we create affordances for the new types of agents without confusing existing game developers. This was also possible because the current view has enough real estate space on the screen that was not being used, as can be seen in Figure 6-7.

6.2.2 Tag Agent

The Tag Agent is an agent with an agent description and a form, and it allows a player to "tag" their current location and leave information about that location behind. The Tag Agent is designed to be accessible at anytime for the player, because a player should be able to initiate the interaction when they find something interesting to tag; instead of placing the Tag Agent at a specific location the map for the player to bump, I added it as a mobile tab to the game. In addition, because a Tag Agent still is characteristic of an agent (in that it has an agent image, name, and description and can be "bumped"), it can be added to the game through the Agent view on the



(a) The popup box for the "New Agent Tab" box.



(b) The popup box for the "Mobile Tabs" box.

Figure 6-8: Popups of settings for the mobile tabs that can be customized in the Settings view. Left is the newly created "New Agent Tab" box, and right is the already existing "Mobile Tabs" box.

online web editor; although it functions differently from regular agents in the game, the separate agent types help capture the distinction.

As seen in Figure 6-7, a game developer can add a Tag Agent from the Agent view. Once the agent has been added to the game, they can then add the tab to the mobile tabs in the game and customize the tab for that agent from the Settings view. A name for the new tab can be given (as shown in Figure 6-8), which will be used as the text on the tab in the mobile app. When a player taps the custom-made Tag Agent tab, the player bumps the Tag Agent and any associated bump scripts with the Tag Agent will be run.

6.2.3 Choice Agent

The Choice Agent is the other variant of an agent with a form. The Choice Agent allows only one submission to their questions. Players can revisit a Choice Agent and change their answers, but only the most recent set of answers is kept track of at any time. Both the Choice Agent and the Tag Agent display similar agent dashboards, but the Choice Agent will be more similar to a regular agent as it is designed to

appear on the map in-game.

Once a player bumps a Choice Agent, the agent will show up in the History mobile tab. From this tab, the player can see the choices they had made, but they will not be able to edit any of their choices. A player may also re-bump a Choice Agent; when this happens, the player will see the choices they had made but will also be able to edit their choices. However, it may be the case that a game developer does not want to allow a player to edit their choices; this can be achieved by not allowing a player to re-bump an agent, which is an already existent feature in TaleBlazer.

6.3 Reviewing Information

In addition to collecting information from players, it would be helpful to display that information back to the user during gameplay. As discussed previously, I designed two agent types with player inputted information - the Tag Agent and the Choice Agent. Other than how the player initiates the interaction and how many times a player can submit, another difference between the two types of agents is what happens to the submitted information for each agent. The Choice Agent was designed more for the sequential decision making use case game, where a player would input information that impacts the gameplay, and not necessarily information that would be useful to review. The Tag Agent was designed more for the data collection use case game, where it makes more sense for a player to review the information they had gathered so far. To simplify the two designs, I have created ways to review information from the Tag Agent and not the Choice Agent.

Once a player submits information through the Tag Agent, they will have two options to review the information they had submitted so far. The first option is through the Tags mobile tab, and the second option is through the Map mobile tab.

6.3.1 Tags Tab

From the beginning, I did have an idea for players to review information that they had collected or inputted through agents with forms. An example mock up can be

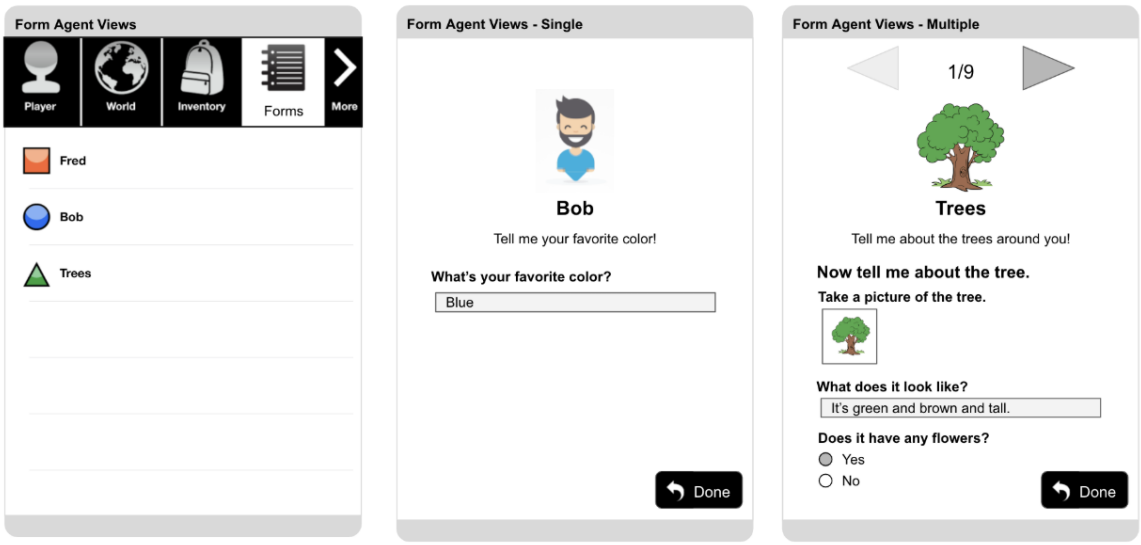


Figure 6-9: Mockup of how players could review information. The leftmost image is a new tab with a list of agents, the center image shows a singular record view, and the rightmost image shows a multiple record view.

seen in Figure 6-9. Note that this was before I had the notion of separating the agents into two different types, so there is no distinction between agents who allow one submission and agents who allow multiple submissions.

After I came up with the notion of Tags, I created a mockup of how this tab might look, as shown in Figure 6-10. It was clear that the location and time of submission information would be important for Tags, so I chose to make those visible when reviewing information, so players could more easily distinguish between different submissions for the same Tag Agent.

Within the Tags tab on the mobile app, there are further two ways to review information. The first is seeing a list of all the tags submitted, ordered by the time of submission. In the case of having multiple Tag Agents (e.g., having a Tag Agent for collecting information about trees and another for collecting information about birds), the tags would be mixed together in this view. The second is seeing a list of Tag Agents, with the submissions grouped by the Tag Agent. When a player taps the Tag Agent, they would see all the submissions for that agent in a carousel view. Figure 6-11 shows the implemented Tags tab with the two different ways to view information.

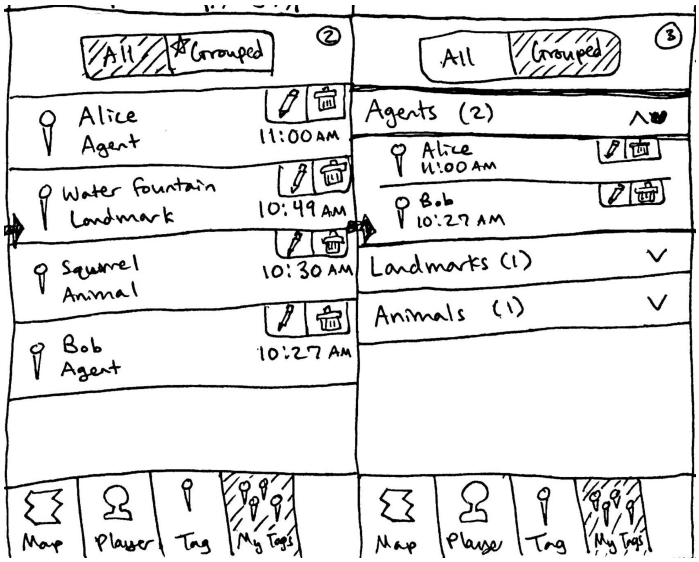
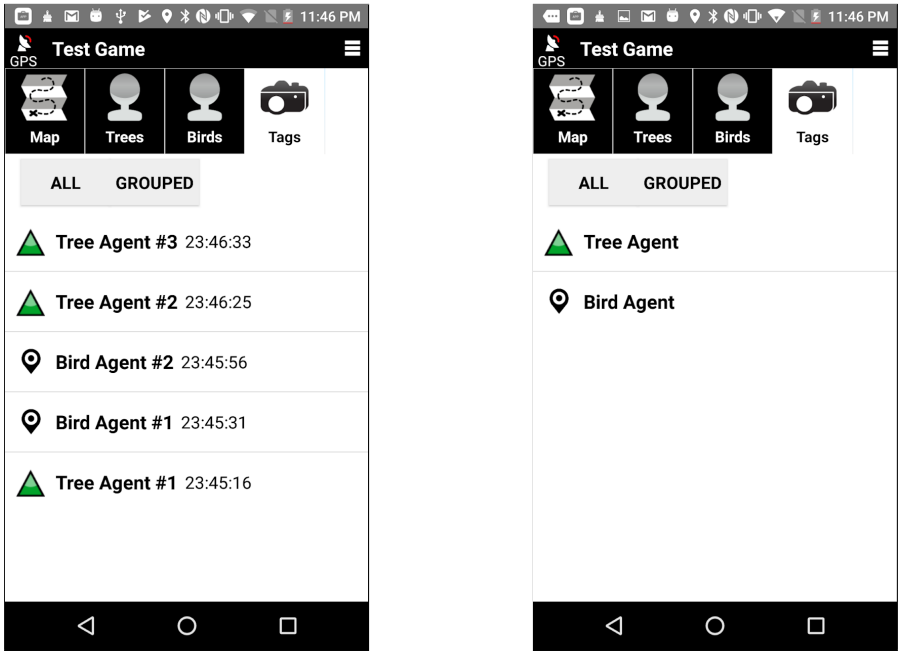


Figure 6-10: Mockup of how players could review information using a Tags tab on the mobile app.



(a) The "All" view of the Tags tab. (b) The "Grouped" view of the Tags tab.

Figure 6-11: Screenshots of the two different views for the Tags tab. Left is the "All" view with the submissions ordered by time, and right is the "Grouped" view with the submissions grouped by the Tag Agent.

When a player is reviewing information through the Tags Tab, the submissions will appear in an uneditable state. However, the game developer can specify whether players can edit submissions, per Tag Agent; if they specify that, the players will also see an edit icon when reviewing the information that will allow them to edit their submissions.

6.3.2 Tags on the Map Tab

Because the Tag Agents are meant to help tag locations, instead of only displaying them in a list view on the Tags tab, it was important to also place the submissions on the Map tab of the mobile app. This makes it easier for players to visually review the locations that they had tagged and where they had seen interesting things. The tags will appear on the map as the Tag Agent's icon, which is chosen by the game developer and which will default to a pin icon (as shown in the icon of the "Bird Agent" in Figure 6-11). To implement this feature, I looked at TaleBlazer's cloning feature, which allows a game developer to clone an agent and have it appear in multiple places on the map. The cloning feature presented a similar functionality by how it creates copies of the same agent in multiple locations. However, as opposed to how the cloning feature allows a player to bump the agent, a player cannot bump a tag on the map; instead, they can tap it to review the information for that particular tag.

If a player submits a tag, the tag will appear at the player's current location. Because TaleBlazer uses different map regions, it was also important to distinguish the tags' behaviors between regions. The tags will appear only on the region the player was in when tagging the location; this makes it simpler on the implementation side because each location has its own x,y coordinates on the map for the current region. If I were to include tags across multiple regions, I would need convert the x,y coordinates for each region the player changes into. This also makes it simpler for players because they wouldn't see tags from a previous region when they change regions.

TaleBlazer also supports indoor games with the use of beacons. Because there is

no equivalent x,y coordinates for indoor games, the tags would not be recorded on the Map tab; however, they can still be recorded and reviewable from the Tags tab.

Chapter 7

Future Work

There are several areas that can be expanded upon following completion of this project. One area for improvement would be giving game developers a simple way to track player choices from Choice Agent submissions. Another area for improvement would be allowing game developers to make use of aggregated information from Tag Agent submissions. The other area would be allowing players to do more with the information they have gathered either in-game or outside of the game. Learning does not have to stop within the game, and with more ways to interact with the information outside of the game, learning can continue after the gameplay has ended.

7.1 Blocks for Choice Agents

A useful feature would be giving game developers a simple way to track player choices. This may be useful if a player's form submission can impact how the game progresses. For example, suppose players are asked to count the number of trees around them, and they are only allowed to move forward in the game if they input that they can see at least two trees. The game would have to be able to know what the player inputted into the form and compare it against a value specified by the game developer.

As mentioned in the previous section, an agent can have actions, which are buttons that the player can press to interact with an agent. Currently, this is the only way for players to influence how a game progresses. However, after a player has made a

choice, there is no way for the game to know what choice was made later on in the game. For example, suppose a player bumped an agent, and the player could choose between talking to this agent or ignoring this agent. And suppose at the end of the game, the player receives a present for having chosen to talk to that agent, or a coal for having chosen to ignore that agent. In order to create an event as simple as this in the game, the game developer would have to create some round-about logic to keep track of the player's state and know what choice the player had made, because there's no built-in way to simply check what choice a player had made during gameplay. A proposed solution for this problem would simply involve creating blocks (as part of the blocks-based programming editor) to test the value of a player's choice. If game developers could more easily incorporate player choices into gameplay, players could have a more meaningful and engaging experience with TaleBlazer, which goes towards our goal of maintaining a fun experience for users.

7.2 Aggregate Information from Tag Agents

One way to expand upon this project might be to help make the Tag Agent submission information more meaningful for the users. Although it is difficult to use the value from any single question from the form, because there would be multiple values to choose from if there are multiple submissions, it might be useful to allow game developers to review and use aggregated information from the forms. For example, suppose there was a TaleBlazer game centered on collecting information about nearby trees, and this game was being used as an educational tool in a classroom. As the students are collecting information about the trees, it might be helpful and meaningful if they could see some aggregate information about all the trees they have seen so far. This information might involve the min, max, average, or sum of all values of a "number"-input type of question, or it could involve getting the count of all submissions the student has made so far. There are many examples of how to make use of collected information from databases, so this work could be inspired by database design.

7.3 Sharing Information

Another area for expansion could be improving upon player-to-player interaction. There are other aspects of TaleBlazer that are being worked upon, namely bluetooth interaction between different players in-game. With this module, it might be possible to build up a set of observations about the world together with another player, being able to modify and influence each others' information.

In addition to in-game sharing, it might be interesting to export or share information from these games outside of the games as well. For example, suppose the previously mentioned example of a TaleBlazer game about observing trees being used in a classroom. After making individual observations, the students could all come back and identify various trends about the trees they observed. They could see how the information they gathered, combined with the information gathered by their classmates, can lead to some interesting conclusions.

Chapter 8

Conclusion

Through the design and implementation of customs forms, Tag Agents, Choice Agents, and new mobile tabs to review information, I believe TaleBlazer will be better equipped to encourage scientific learning through its games. I created these features to help enhance the two main end use case games mentioned before - the Data Collection game and the Sequential Decision Making game. The main goals were to design intuitive, useful features that can be used for a variety of different ways, and that can enhance both the education and the fun of TaleBlazer games.

In addition to the implementation of the features, there were some key takeaways I learned from simply working with the TaleBlazer team. One key lesson was in widening my perspective of who I was designing for. Often times, I found myself thinking I had a good solution but realized there were a lot of different types of users I had not considered when coming up with the design. For one thing, TaleBlazer users have a wide breadth of experience, so I had to make sure new features were obvious and intuitive while also adding enough depth to the feature to make it usable for experienced users. In addition, because TaleBlazer is already being used, I had to ensure each new feature I implemented lets current users continue using the platform effortlessly and does not break any current games. Another key lesson was working with a large codebase that has been and probably will continue to be modified by a lot of people. There were challenges in understanding the code from a perspective of needing to modify it, but there was also a challenge in making sure my code would be

understandable and easily modifiable by the next person who uses it. To this end, I used existing classes and incorporated my code into existing pipelines where possible.

Appendix A

User Testing Documentation

A.1 Background

Hi, thank you so much for coming to visit us. My name is Nayoung [and intro myself a little]. Today, we're pilot-testing a new feature, so any problems you run into will help us improve the software and are not a reflection of your capabilities. So please don't hesitate to let us know if you run into any questions or problems at any point.

Just to give you a little background, TaleBlazer is a platform for creating location-based AR games (mention Pokemon Go). Anybody can create a game on our online editor at taleblazer.org, and then you just download the mobile app and start playing! So I'll briefly talk about these two components: the app and the web editor.

In the app, you'll see a map and a small blue circle for your location. The other icons on the map are what we call agents, which are characters in the game. When you walk close enough to an agent's location, you'll bump into the agent and see its information and other ways to interact with the agent.

JUDY can show quick demo of Grapefruit Tour. (PREP on 2 devices)

So all that is on our mobile app, which is how people play TaleBlazer games. The games are created on our web editor (usually this is done on a computer). You can choose the map location, create new agents, add tabs to the game, and customize other settings such as how agents are bumped. Then you can also add logic to how the game is played, by using blocks (mention Scratch or AppInventor as examples).

Right now, we're working on allowing players to input their own information into TaleBlazer. Things like having the agent ask for a player's name or a player's favorite color. So we'll have you test out what it might look like to try to create an agent that can do that. We have a pretty wide audience, ranging from novices to experts, and our goal is to make this system easy to understand for beginners while also useful for more experienced users as well.

Do you have any questions so far?

Now I'm going to explain how this will work. We'll be using something called the think aloud model today. I'll give you guys some tasks to do and I want you to think aloud any thoughts or actions that you're doing as you're doing them, anything that comes to mind. This is really helpful for us because we learn more about things that seem off or things that seem to work well from just how you're thinking. [talk through example of making PBJ sandwich]

Now I'm going to list out several tasks for you to do, and I would like you to try to do them while talking through your thought processes as much as possible. Also, let me know if you need to take a break at any point.

A.2 Tasks

1. Brief Instructions on Getting Started
   (a) Go to dev.taleblazer.org
   (b) Go to the Agents tab
2. Create a new Form Agent. This agent will be asking the player for information.
3. Give the new Form Agent a name and description. [If they can't find it – let them fail, tell them it's somewhere on this page, show them where it is]
4. Ask the player for the following:
   (a) Their name
   (b) Whether they like dogs or cats better

(c) What their favorite season is (a choice from spring, summer, fall, winter)

(d) Their age

(e) One photo of himself/herself

5. Make the question for their name required.

6. Order the questions as follows:

(a) Name

(b) Photo

(c) Age

(d) Dogs or cats

(e) Favorite season

7. Delete the question about dogs or cats.

8. Change the questions about seasons to allow the player to choose multiple favorite seasons.

9. Undo the last two actions (changing the seasons question and deleting the dog/cat question).

A.3 Follow-Up Questions

1. Was there anything that was frustrating or confusing about what you just did?

2. What did you like the most? What did you like the least?

3. Have you ever used any other technology to create a form or a survey?

4. Did you know what all the question types meant? Or are there different wordings that would have made more sense?

5. Do you have any suggestions?

6. Do you have anything other thoughts you'd like to share?

Bibliography

[1] United State Environmental Protection Agency. Basic information about citizen science, 2019.

[2] Janis L. Dickinson, Jennifer Shirk, David Bonter, Rick Bonney, Rhiannon L. Crain, Jason Martin, Tina Phillips, and Karen Purcell. The current state of citizen science as a tool for ecological research and public engagement. *Frontiers in Ecology and the Environment*, 10(6):291–297, August 2012.

[3] John Losey, Leslie Allee, and Rebecca Smyth. The lost ladybug project: Citizen spotting surpasses scientist's surveys. *American Entomologist*, 2012.

[4] The Lost Ladybug Project. Recent publications from the llp, 2018.

[5] Elizabeth R.Ellwood, Theresa M.Crimmins, and Abraham J.Miller-Rushing. Citizen science and conservation: Recommendations for a rapidly moving field. *Biological Conservation*, 208:1–4, April 2017.

[6] Jessica Sickler and Tammy Messick Cherry. Lost ladybug project: Summative evaluation report, 2012.