TaleBlazer Documentation

How to Read this Documentation

TaleBlazer specific terminology is denoted with italics. Example game functionality which is not intrinsic to the TaleBlazer software is denoted via single quotes. Thus a game designer might configure a 'Time Machine' *agent*. In this case, 'Time Machine' denotes a specific example of an agent that the designer might create, while *agent* denotes the TaleBlazer term for an object that the game designer creates on the map.

Overview

TaleBlazer is a location-based augmented reality game platform. The term *augmented reality* means that the game play takes place in a real world context and not merely on a computer screen in a classroom or at home. The game *augments* the real world context by providing information and game play mechanics that are in some way connected with the player's interaction with the real world. The term *location-based* means that the connection between the virtual world and the real world is driven by the player's location in the real world as determined by a GPS signal. Playing a TaleBlazer game, therefore, involves walking around in the real world with the TaleBlazer software running on a GPS enabled device. As the player moves around in real space, their GPS location allows them to interact with "nearby" virtual objects in the TaleBlazer game world using the mobile software.

To make a TaleBlazer game, the game designer specifies one or more *regions* – real-world locations where the game takes place. Then the game designer adds *agents* – representing items, characters, or whatever the designer wants – to the region. When the player's location comes close enough to the agent's location, the player is said to *bump* into the agent. When bumping an agent, the *agent dashboard* comes up by default, displaying the agent's name, description and image. The designer can also specify *traits* (variables) displayed as a list and *actions* displayed as a row of buttons on the agent dashboard.

A Note on GPS - Using the location-based features of TaleBlazer requires to the game to be played outdoors. Indoor locations do not have access to a valid GPS signal. Outdoor locations can have weak or poor GPS signals, particularly near tall buildings, open areas of water, and in sparsely populated areas. Even under best case conditions, GPS positioning is only accurate to about 3 meters. The game designer can compensate for inaccurate GPS signals by adjusting the *bump settings*. See the section on *Bump Settings* later in this document. There are other considerations to keep in mind when designing a location-based game such as pedestrian accessibility/safety, etc. TaleBlazer also supports *Indoor* regions, which do not require a GPS signal. See the section on *Indoor Regions* later in this document.

More functionality can be achieved by configuring *scenarios* and *roles*. Just as the player can interact with the agent via the agent dashboard, the player can also interact with the world dashboard, the player dashboard, and the team dashboard (coming soon to multi-player). Complex game mechanics can be specified by means of a blocks-based programming language

TaleBlazer consists of four software components: an online game *Editor* which allows the designer to create/edit/save games, a game *Repository* server which stores the games, a multi-player server (currently under development) which maintains a shared game universe for multi-player games, and an installed mobile application which is used to play the games on iOS or Android devices.

Editor Software

The editor software is used by the game designer to create a TaleBlazer game. Across the top of the page are the tabs that organize the game content and the game control panel. Use the buttons in the control panel to *Save* or create a *New Game*. Note that TaleBlazer does not automatically save, so save early, save often! A drop-down from the game control panel allows the game designer to modify the name, image, and description of the game. These settings are visible on the *game page* when the player chooses to start playing the game.

The tabs separate the content into five areas, in more detail below.

World

The world tab includes all the settings that are relevant to the entire game world:

Mobile Tabs

The game designer specifies which tabs are visible in the mobile software when the game is played:

- **Game** – displays meta information about the game itself such as the name, image, and description of the game, the game code, as well as a *Leave Game* button which closes the game and turns off the GPS.
- **World** – shows the world dashboard, including name, description, image, as well as world traits and actions.
- **Map** – shows the current location of the player as an icon on the game map along with the icons for the visible agents. If the player is close enough, as specified by the *bump settings*, the player can tap on agent icons to bump the agents.
- **Player** – shows the player dashboard, including the name, description, image, as well as player-specific traits and actions.
- **Clue Code** – here the player can enter a *clue code* which allows the player to interact with a *clue code* agent 'anytime, anywhere' – ie regardless of the player's GPS location
- **Heads Up** – uses the compass and GPS readings of the device to show nearby agents as markers overlaid onto the video camera display. If the player is close enough, as specified by the *bump settings*, the player can tap on these markers to bump the agents.
- **Inventory** – shows the contents of the player's inventory
- **History** – shows a list of agents that the player has previously encountered in the game
- **Log** – shows a list of all the things the player has done in the game.
- **Settings** – allows the designer to access debugging functionality such as turning ON tap to visit, and changing the bump radii

Multi-player vs. Single Player

(Under development)

A multi-player game includes multi-player game dynamics and requires the players to have consistent Internet access via a data plan or Wi-Fi signal to communicate with the multi-player server. Players in multi-player games experience a single shared game world in which changes to the virtual world are propagated to all players playing the same game. For example, if one player picks up an agent, then that agent will disappear from the map for all other players. Multi-player features include:

- Players experience a shared game world.
- Players can pass objects to each other.
- Players can engage in real-time in game chat with other players.
- Players can experience team-based game dynamics which require a *team* to accomplish goals together. Team-based dynamics are configured by the game designer using the *Teams* tab and certain team-based blocks in the blocks-based programming language. For example, the team may be required to have certain objects in their collective inventories in order for a new agent to become visible on the map or in order for the players on the team to be able to progress to the next region.

Introduction

The *introduction* comes up when the player first starts to play the game. The game designer can use the introduction to orient the player to the game in some way - set up the narrative, give the player a quest, or provide help on how to use the software or how to play this game.

Mobile Settings

Certain advanced settings for the mobile software can be found here. The game designer can specify which key pad is presented to the player for entering passwords and clue codes, and whether the player must enter a password to show the *Settings* tab. The TaleBlazer mobile software allows the player to turn on *tap to bump* via the *Settings* tab. *Tap to bump* allows the player to tap on the map icons to visit the agents regardless of the player's location. This is important functionality for the game designer to be able to test their game when not on location. The savvy TaleBlazer mobile user can include the *Settings* tab for any game by tapping the *Show settings* checkbox in the *About* box (accessible by tapping the *About* button on the *Game* tab. Savvy game designers can protect this functionality by designating a password so that players cannot turn on the *tap to bump* setting unless they have the password.

Bump Settings

These game-wide settings allow the designer to control how hard or easy it is for a player to *bump into* an agent. The designer specifies

- Whether *bumping* into an agent
  - shows the agent dashboard AND runs the *bump script*(s) OR
  - just runs the *bump script*
- How close the player must get to the agents in order to *bump* into them.
- Whether the agents can be *re-bumped*
  - How far the player must get from the agents in order to allow them to be *re-bumped*
  - By default, once an agent has been encountered, the player does not *re-bump* the same agent later in the game. If the *re-bump* option is checked, then agents can be *re-bumped* if the player leaves the immediate area and returns. This distance is enforced to prevent the agent from continuously popping up when the player is standing in one place.
- Whether the agent's map icon is
  - Hidden on the map
  - Visible on the map
  - Only visible when the player is within a specified distance
- Whether the player can *bump* the agent by tapping on its map icon in the map view.
  - Regardless of how far the player is from the player
  - Only when the player is within a specified distance from the agent
- Whether the agent's marker in the Heads Up View is
  - Hidden
  - Visible
  - Only visible when the player is within a specified distance
- Whether the player can *bump* the agent by tapping on its marker in the Heads Up View.
  - Regardless of how far the player is from the player
  - Only when the player is within a specified distance from the agent

A Note on GPS Accuracy – for very small regions, the imprecise nature of the GPS signal will be more readily apparent to the player. The player icon will appear to jump around erratically and if the bump setting is configured in game coordinates, the players may have a very difficult time getting their player icon to get close enough to the agent icon to bump the agent. Mitigate this issue by either using meters instead of game coordinates and/or by increasing the threshold to allow players to bump into agents when further away. The player may still see their icon jump around erratically, but it won't be as hard to bump into the agents. Alternatively, you can consider including a larger region than absolutely necessary.

Agents can be configured individually to behave differently from each other. See *Bumping Agents* under the *Agents* section later in this document.

Scenarios

Scenarios allow the game designer to specify different 'versions' of the same game that the players can pick from when they start the game. For example, the player might be asked to pick between 'Easy' and 'Hard' or between 'Short' and 'Long' or even 'Start at Main Entrance' and 'Start at Side Entrance'. Think of a scenario as a multiple choice question that the player answers at the beginning of the game. The game designer can use the player's 'answer' (ie which scenario they picked) in the game logic (as specified by the blocks-based programming language) at any later point in the game to take into account the player's choice.

Actions

Actions appear as buttons on a dashboard. Actions can be:

- **Text** –displays rich formatted text
- **Video** – plays an uploaded video or (under development) a YouTube video
- **Built-in** – performs built-in functionality, such as the *pick up* and *drop* actions.
- **Script** – executes a designer created script. See the section on *Scripting* later in this document.

The name, type, contents, and visibility of an action can be changed in the Editor via the relevant *Actions* box. The name of the action appears as the text on the action button. The visibility of the action determines whether the button will be visible to the player on the relevant dashboard. The designer can also specify the sort order of the buttons across the dashboard. The visibility of an action can also be changed at run-time (during game play) by means of the blocks-based programming language.

A Note on Shared Actions – the designer cannot choose to share actions among multiple agents, teams or roles. Only the *built-in* actions are shared. Changing the name of the *pick up* or *drop* action will change it for all agents. All other actions appear on the dashboard of only a single agent, team, or role.

To change which actions are visible on the dashboard during game play, add one or more *hide/show action* block(s) for example, under a *when player bumps* block (to hide/show agent actions) or add one or more *hide/show action* block(s) under a *when player sees player/world/team tab* blocks (to hide/show other actions).

Traits

Traits are variables that the game designer can specify for agents, roles (ie players), teams, and for the world itself. The game designer declares and initializes a trait by pressing the *Add Trait* button and then using the *trait dialog* to add a trait to the given object (agent, role, team, world). When adding a trait, the designer specifies the name, the initial value, and the initial visibility of the trait. The visibility of the trait determines whether it will be visible to the player on the relevant dashboard. The name, value, and visibility of a trait can be changed in the Editor via the relevant *Traits* box, as well as the sort order of the traits. The value and visibility of a trait can also be changed at run-time (during game play) by means of the blocks-based programming language. When creating a new trait the designer must also specify the following attributes of the trait that cannot later be modified:

- **Scope** – which object(s) can use this trait (not applicable to world traits since there is only one world) -
  - Common for all objects of the same type (agent, role, or team)
  - Only for the current object (agent, role, or team)

World traits are global settings shared by all players in a game. For example, the designer can use a world trait to specify and display the current temperature in the virtual world, or the DOW Jones Industrial index, or the current level of the terrorist alert. We recommend that all player traits be created as common traits. This will allow you maximum flexibility in accessing the traits from all scripts in your game.

A Note on Creating Traits – In single-player games, there is no logical difference between world, team, and role traits. Each player has separate instances of all world, team, and role traits. Single player games run independently on each player's device without sharing a game world. However, in a multi-player game, world and team traits are shared among all players in the same world or team respectively because they are part of the same shared game world. For consistency's sake, even in single player games, the designer should create player variables as role traits and game-wide (global) variables as world traits. For example to keep track of the player's score, the game designer should use a role trait and not a world trait.

To change which traits are visible on the dashboard during game play, add one or more *hide/show trait* block(s) under a *when player bumps* block (to hide/show agent traits) or add one or more *hide/show trait* block(s) under a *when player sees player/game/team tab* blocks (to hide/show other traits).

Scripts

The rest of the page is taken up by the script editor. For more information on scripting, see the section titled *Scripting* later in this document.

MAP

On this tab, the game designer specifies a location in the real world where their game will take place. In order to play the game, the player needs to physically be in this location. The TaleBlazer *Map* tab shows a Google Maps view of the world which can be searched via a search bar. The designer can pan and zoom this view, and then can use the *Move Game To Here* button to move the *game boundaries* to the area visible in the map view. Alternatively, the designer can change the game boundaries by clicking and dragging on the blue square markers in the map view or by typing new lat/long values for the boundaries in the property box on the left side of the screen. To prevent the unwary designer from accidentally changing the game boundaries, the map is *locked* by default once the boundaries have been set. To change the boundaries of the game at any time after this point, unlock the *map* by unchecking the *Lock Map* checkbox.

The agents are visible in the map view as map icons. The game designer can edit the agents by clicking on them or (if the *Lock Agents* checkbox is unchecked) can move them by dragging. To keep the agents in their real world location and tweak the boundaries of the map – perhaps to include a larger or smaller area or to move the map slightly – use the *Preserve Agents' Lat/Lng* setting. To move the entire map to a new place, use the *Preserve Agents' X/Y* setting.

The game designer may also want to specify more complex map features:

Custom Map

By default the mobile software uses the Google Maps API to display the player's position in the real world during the game. This API requires the player to remain within Wi-Fi or cell tower coverage to display the game map properly because the map is updated dynamically during game play as the player moves about the real world. In order to create a game that can be played without a data plan or in an area with poor Wi-Fi and cell tower coverage, the designer can upload a *custom map* – a jpg or png file which is displayed on the background during game play in lieu of the Google Map.

A custom map can also be used to show custom details on the map during game play, such as pedestrian paths, enhanced color, historical or geographic data, or even a fictional landscape.

How to Capture a Satellite Image of the Game Area

On the *Map* tab, pick the relevant region, then press the *Capture Image* button in the *Map Settings* section. The next step is to download the captured image to your local hard disk by right-clicking (or control clicking for the Mac) on it and selecting *Save Image As…* from the context menu. Once you have downloaded the captured image, you can open it in an image editing application and add layers for the buildings and pedestrian paths. You can even add text to give the player instructions. Once you have the desired image, upload it by pressing the *Upload Image* button.

Multiple Regions

Multiple *regions* can be used to define different physical locations in which to play the game, such as two different areas of a school campus. But regions can also be used to create different layouts on the same physical location. The location of a TaleBlazer player or agent is specified by coordinates and by region. The map view on the handheld only shows the players and agents that exist in the same region as the player. For example, you can use multiple regions each mapped to the same real world location to represent different game levels, time periods, or different outcomes for the player's in-game decisions. All players start in the same *default* region, specified via a checkbox on the Maps tab. There is no automatic way for a player to move from one region to another – the game designer must provide a way for the player to change to another region via a *move to* block in a script. For example, the game designer can create a 'Time Machine' *agent* that includes a 'Visit the Future' *action* which calls a script that includes a *switch region* block. Thus the player can get to the 'Future' region by meeting the 'Time Machine' agent and tapping on the 'Visit the Future' action. For more information on this example specifically and for more information on agents and actions in general, see the section on Agents.

Indoor Regions

Indoor regions provide a way to include game play in indoor spaces if part of the game will be played indoors out of range of a GPS signal. The software does not detect when the player moves indoors or outdoors. If the player's in game location is set to an *outdoor region*, the GPS device will continuously attempt to find a signal, even if the player moves to an indoor location in the real world. Likewise regardless of the player's real world location, when a player's in-game location is set to an *indoor region* via the *move to* block, the GPS device is turned off and no player icons appear on the map. Instead of bumping into the agents via the GPS signal, the player bumps into agents in an indoor region by tapping on them. Indoor regions are often used with *password protected agents* to force the players to walk around the physical space. See the section on *Password Protected Agents* later in this document.

Agents

The game designer creates agents in a region and gives each agent a name, description, and image. The designer can also specify the agent's icon, which is displayed when the agent is visible on the map tab and alongside the agent's name in other places in the mobile software.

Settings

Clue Codes

Agents can be configured to be located as a *clue code* instead of at a coordinate in a specific region. A clue code agent is bumped when the player types in the correct clue code on the clue code tab. The game designer can use clue codes to implement the same functionality as an audio tour in a museum in which the player searches for signage and then types in a number. Clue code agents can be accessed anytime, anywhere.

Password Protection

The designer can opt to password protect the dashboard so that the player must enter a password in order to see the traits and actions on the agent's dashboard.

A Note on Clue Codes vs. Password Protection – Password protection differs from clue codes in the way that the agent is accessed. With a password protected agent, the agent still has a location in the real world and its dashboard comes up automatically when the player *bumps* into it. The player has an automatic in-game prompt at that physical location that they need a password. For a clue code, the player can enter the clue code to access the agent anytime, anywhere. The game designer must provide the prompt for the player to know when to enter a clue code. Perhaps the introduction tells the player to enter a clue code; perhaps players are told to look for clue codes on signs in the physical world; perhaps an agent in the game tells one player a clue code and they must share it with the other players; perhaps the narrative of the game allows the player to interact with the same clue code agent differently at different points in the game.

Excluded Agents

Excluded agents are inert and inaccessible by the player. When an agent is excluded, players cannot see the agent, *bump* the agent, or interact with it. An excluded agent, however, can be referenced by the scripting language. The traits of an excluded agent can still be accessed and/or changed via scripting. An excluded agent can be included via the *include agent* block; likewise, an included agent can be excluded via the *exclude agent* block.

Bumping Agents

When the player encounters or meets an agent in the TaleBlazer software it is called *bumping* the agent. *Bumping* an agent can happen in one of five ways:

1. GPS proximity - when the player is "close enough" to the location of the agent
2. Map icon tap – when the player taps on the agent's map icon in the *Map* view
3. Heads Up tap – when the player taps on the agent *marker* in the *Heads Up* tab
4. Inventory tap – when the player taps on an agent in their inventory tab
5. Clue Code - when the player enters the correct *clue code* (for 'clue code' agents only) on the *clue code* tab.

When the player *bumps* the agent, the agent *dashboard* pops up by default, showing the agent's name, description, image, and visible traits and actions.

Note on History – the history is meant to provide a way for the player to review information they've already seen. The player can review the traits of the agent at the time they last visited the agent and can see the *text* or *video* actions that were visible when they last saw the agent, but the player cannot see the current values of the traits nor any actions that might change the game play such as *built-in* actions or *script* actions. When the player views the agent dashboard via the history tab, the bump script does NOT execute.

Bump Settings

Certain bump settings can be configured differently for individual agents. By default, each agent is set to use the game-wide settings. To create an agent that can only be seen in Heads Up view, uncheck the *Use game settings* for that agent, then uncheck *Allow GPS proximity to bump this agent,* check the *Hidden on map* checkbox, and check the *Visible in Heads Up view* radio button and the *Allow player to tap to bump* checkbox. The agent will not be visible or accessible in the map view, but will be visible and *bumpable* in the Heads Up view.

Actions

See the *Actions* section under the *World* area earlier in this documentation.

Traits

See the *Traits* section under the *World* area earlier in this documentation.

Scripts

The rest of the page is taken up by the script editor. For more information on scripting, see the section titled *Scripting* later in this document.

Roles

The game designer can configure a single role for a game, in which case all players experience the same game, or the game designer can configure multiple roles, in which case the game designer can use the scripting language to specify different interactions for players playing different roles. Like an agent, the role can have a name, description, and image associated with it. The player picks the role when starting a single player or joining a multi-player game. Also like an agent, the role can have traits and actions associated with it. During game play, the player can access the role specific functionality via the *Player Dashboard* which is a tab on the main interface of the mobile software. The Player Dashboard shows the name, description and image for the role along with the visible actions and traits.

Actions

See the *Actions* section under the *World* area earlier in this documentation.

Traits

See the *Traits* section under the *World* area earlier in this documentation.

Scripts

The rest of the page is taken up by the script editor. For more information on scripting, see the section titled *Scripting* later in this document.

Scripting

Please see the tutorials available on the TaleBlazer website for more information on scripting while this section of the documentation is under construction.

How to Play a Game

Name your game so you will be able to identify it again easily. Then click the *Save Game* button. On the device, find the game either by logging into your account and tapping on it in the *My Games* area or enter the game code. Once you have found your game, tap on the *Download and Play* button. To invite others to play your game, give them the game code.