User's guide

Authors:
Víctor M. Pérez Colado,
Iván J. Pérez Colado,
Cristina Alonso Fernández,
Ana Rus Cano,
Leandro Flórez Aristizábal
Rosa Peromingo Guzmán,
David Pérez Cogolludo,
Alma Gloria Barrera,
Manuel Freire Morán,
Iván Martínez Ortiz,
Baltasar Fernández Manjón

Latest version: 1.6
Updated: September 2020

http://u-Adventure.e-ucm.es

License

<u-Adventure> is freeware: can be used, redistributed, integrated in your Project (event for commercial purposes) and/or modified under the terms of the *GNU Lesser General Public License*, published by the *Free Software Foundation*, either version 3 or newer.

<u-Adventure> is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. The full license terms are available at: http://www.gnu.org/licenses/lgpl.html.

This document is registered in Safe Creative (http://www.safecreative.org) under the terms of license *"Creative Commons Attribution-NonCommercial-NoDerivs 3.0"*. A full version of this license can be obtained at: http://creativecommons.org/licenses/by-nc-nd/3.0/legalcode

According to this license, this user's guide can be shared (copied, distributed and transmitted) under the following conditions:

- Attribution: You must attribute the work in the manner specified by the <e-UCM> group, form Universidad Complutense of Madrid (the author) (but not in any way that suggests that they endorse you or your use of the work.
- Non-commercial: this user's guide can't be used for commercial purposes.
- No Derivative Works: You may not alter, transform, or build upon this work.

Acknowledgements

<u-Adventure> version 1.0 has been partially funded by the following institutions:

- Spanish Ministry of Science, through the national research project TIN2017-89238-R.
- European Commission RAGE H2020-ICT-2014-1-644187, BEACONING H2020ICT-2015-687676, Erasmus+ IMPRESS 2017-1-NL01-KA203-035259.
- Complutense University of Madrid, (research group nº 921340).
- Regional Government of Madrid, through the eMadrid P2018/TCS4307.

The <u-Adventure> platform user's guide is based on the <e-Adventure> platform user's guide credited for the <e-UCM> Team coordinated by Javier Torrente, Ángel del Blanco, Ángel Serrano, Eugenio Marchiori and Pablo Moreno.

Index of contents

Index of figures

http://u-Adventure.e-ucm.es

1. <u-Adventure> basics

1.1. About <u-Adventure>

<u-Adventure> is a platform for the development of classic adventure computer games with educational purposes (although other types of 2D games can also be produced). This platform was developed by the <e-UCM> research group (http://www.e-ucm.es) at the Complutense University in Madrid (Spain). ""Graphic Adventure Games" (sometimes referred as point-to-click adventures)" (sometimes referred to as graphic or *point-and-click* adventures) include games such as the *MonkeyIsland*™ or *Myst*™ sagas, original titles of the genre. The platform is specially focused on this genre because it is considered one of the most suitable for educational purposes.

<u-Adventure> is the successor of the <e-Adventure> platform (http://e-adventure.e-ucm.es). In contrast to its predecessor that was built on Java framework <u-Adventure> runs on top of Unity game engine, that provides graphical engine, multiplatform support and editor tools among other things. Hence, <u-Adventure> is distributed as a Unity game engine extension in the Unity Asset Store and under a unitypackage format. Despite of this, no previous knowledge of Unity is required outside of what can be found in this manual.

The project creation and management is, therefore done by Unity. After a project is created, the <u-Adventure> package has to be imported using the unitypackage file or directly downloading it from the Asset Store. All the information about project creation and management will be further explained in this document.

Previous <e-Adventure> projects can be upgraded to be used in <u-Adventure>. However, <u-Adventure> is not completely backwards compatible, as some features have been deprecated or are still work in progress.

<u-Adventure> 1.0 is the latest version of the platform currently available. It can be downloaded at http://u-Adventure.e-ucm.es. Also, sample games and more information regarding the platform can be found in http://e-adventure.e-ucm.es.

Every <u-Adventure> distribution includes both the game engine and the editor. The game editor is the toolkit that allows for the creation of games and the game engine is the set of Unity components and extensions that will execute the game. Also, a game emulator is included to run projects. Nevertheless, games exported using Unity will not need any of the components to run standalone.

<u-Adventure> requirements are limited to Unity game engine requirements, which runs on both Windows (7 SP1+, 8, 10, 64-bit versions only) and Mac OS X (10.9+). More information about Unity requirements can be found at: https://unity3d.com/unity/system-requirements. The recommended Unity version to run <u-Adventure> is **2019 LTS** (2019.4.X) that can be download from https://unity3d.com/es/unity/qa/lts-releases. Although newer versions might be compatible, unexpected bugs might happen.

If you find <u-Adventure> of your interest, and want to find out more about our publications and other investigations of our research group, you can visit our web site: http://www.e-ucm.es.

1.2. About this document

This document aims to be a complete user guide, including descriptions of every feature available in <u-Adventure>. Although features of the engine and editor are described, this document focuses on the use of the editor for the creation of new educational video games. We tried to cover every available feature, but some of them might be left out. We apologize for any inconvenience.

This is not the only documentation available about <u-Adventure>. You could find shorter

http://u-Adventure.e-ucm.es

tutorials and other learning materials on our website (http://u-adventure.e-ucm.es). You can also access the contextual help pages that are embedded in the <u-Adventure> editor application. Just click on any info button like this and context-sensitive information will be displayed.

This guide is prepared to be read from start to end, taking the reader though the different parts of the editor in an order that can be easily followed and that usually goes from the easiest to the hardest details. However, this guide can also be used as reference and for that a detailed table of contents is included which allows for a quick solution of doubts about the platform.

1.3. Installation and configuration

<u-Adventure> is a framework built on top of Unity distributed in form of Unity extension. Because of this, no installation is required for <u-Adventure>. In contrast, <u-Adventure> has to be deployed inside of each project by using its "unitypackage" or through the Unity Asset Store. In this section, we will explore both Unity installation/project management and <u-Adventure> deployment and project setup.

1.3.1. Installing Unity

Unity game engine can be downloaded from https://store.unity.com/download. Unity personal license allows free Unity usage for individuals and small teams with an annual revenue lower than $100k. For larger revenues, other licenses should be obtained such as Plus, Pro or Enterprise. All the information about Unity licenses can be found at https://store.unity.com/.

Unity installation is done using a step by step wizard that automatically downloads all the desired features (Figure 1). Detailed information about installation can be found in the Unity manual[1]. We want to highlight the Unity installer section for the platform support. That section determines which platforms are going to be installed and therefore will be available for exportation. Figure 1 shows this section of the installer. The components have to be selected by clicking in the checkboxes. <u-Adventure> simplified builder (explained in 1.4.4) allows to export into Windows, Mac, Linux, Android, iOS and WebGL. Therefore, all these platforms are recommended to be installed at this point (although some of them might not be necessary, depending on the project's purpose). uAdventure games can be exported in any of the Unity compatible platforms (using default Unity build pipeline[2]), however no support is given for platforms not supported by the simplified builder.

[1] https://docs.unity3d.com/Manual/InstallingUnity.html

[2] https://docs.unity3d.com/Manual/PublishingBuilds.html

http://u-Adventure.e-ucm.es

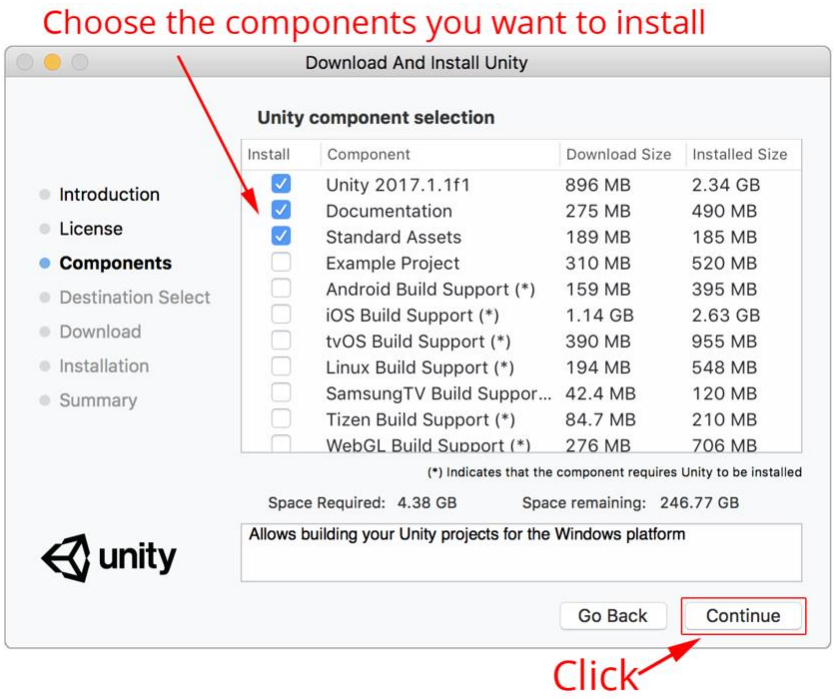


Figure 1. Unity installer platform selection. From Unity3d.com

Once Unity is installed, when opening it, it will show the Unity log-in screen (Figure 2). If you already have a Unity account you can use it now entering your email and password. Otherwise, click on the "create one" blue text to create a new account. You can also log-in using one of the social logging buttons for Google or Facebook.

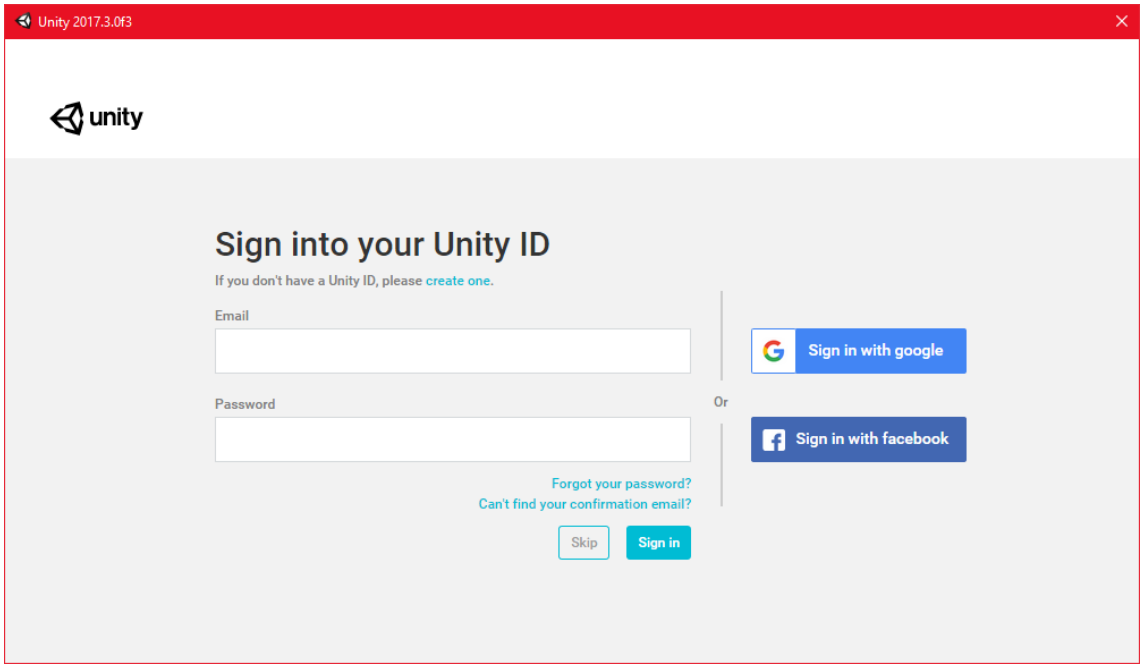


Figure 2. Unity log-in screen.

Once the account is created or you are logged, you can select the license. However, it is possible that a survey of usage is prompted when selecting the Personal license. Complete it as required and then you will see the Unity project management window (Figure 3).

http://u-Adventure.e-ucm.es

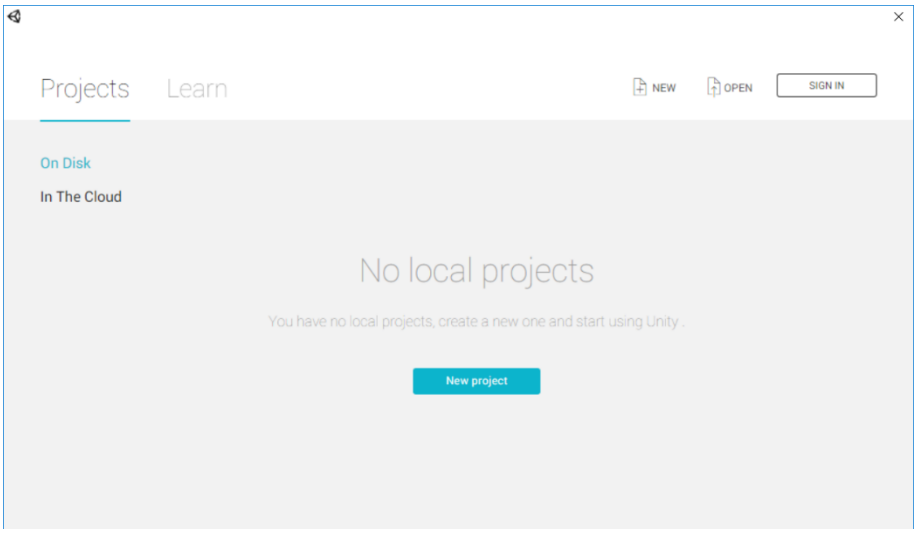


Figure 3. Unity project management window

In the next section, we will use this window to create our first Unity project and we will learn how to import <u-Adventure> inside the project.

1.3.2. Creating a Unity project

In the Unity project management window shown in Figure 3, we will be able to create and open Unity projects and therefore, our previously saved <u-Adventure> projects. To create a new project, we will click on the "**New**" button in the top-right corner Figure 4.

Once clicked we will have to introduce a project name and select a location for the project. Since we are not going to use the Unity editor features, it does not matter if we select 2D or 3D. Finally, we recommend disabling the Analytics (Figure 4) unless you specifically want to use Unity analytics. More information about Unity Analytics can be found at the webpage[3].

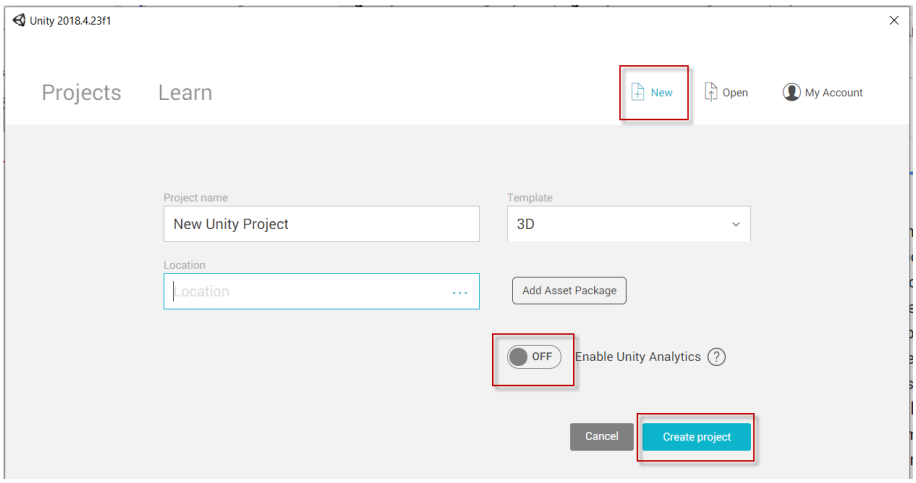


Figure 4. Unity project creation.

Finally, we will click on the "**Create project**" button below and, once the progress has finished, we will see the Unity editor in its default layout as seen in Figure 5. The position of the different editor parts (scene viewer, project management, hierarchy, etc.) might vary depending on your settings.

[3] https://unity3d.com/solutions/analytics

http://u-Adventure.e-ucm.es

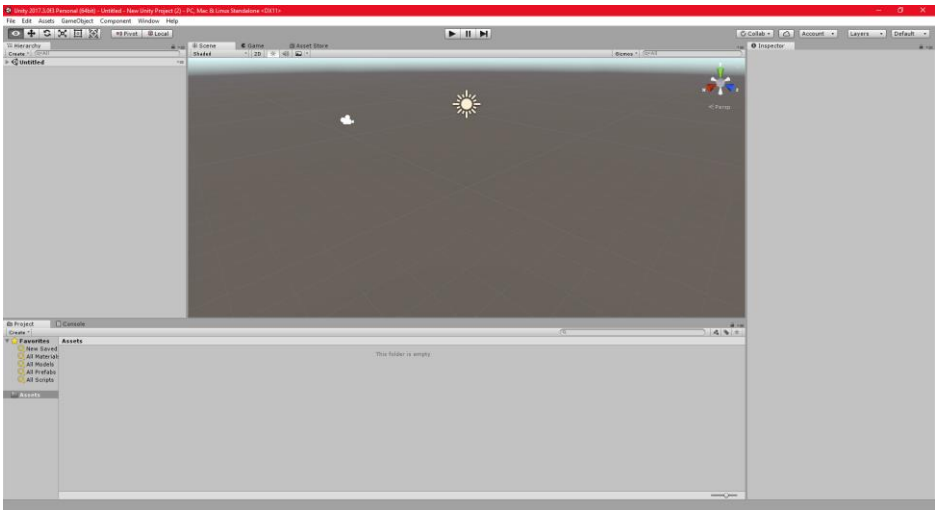


Figure 5. Unity editor with default layout.

1.3.3. Importing <u-Adventure> in our project

Once the project is created, we will import <u-Adventure>. **This step must be done in every new (blank) project.** However, for previous <u-Adventure> projects opened in other computers it is not necessary to re-import the package, as it is already included inside the project.

There are two possible ways to import <u-Adventure>: 1) download the package and import it manually; or 2) install it through the Unity Assets Store. You can choose any of the methods as only one is enough to set up <u-Adventure>.

1.3.3.1 Using the package

The package is a file that contains all the <u-Adventure> scripts and assets. To get it, it is possible to download it from the <u-Adventure> website or from the uAdventure GitHub project (see "Releases" section[4] Figure 6). The file is identified by its extension "**unitypackage**" and, once downloaded, the Unity logo will appear as the file icon.

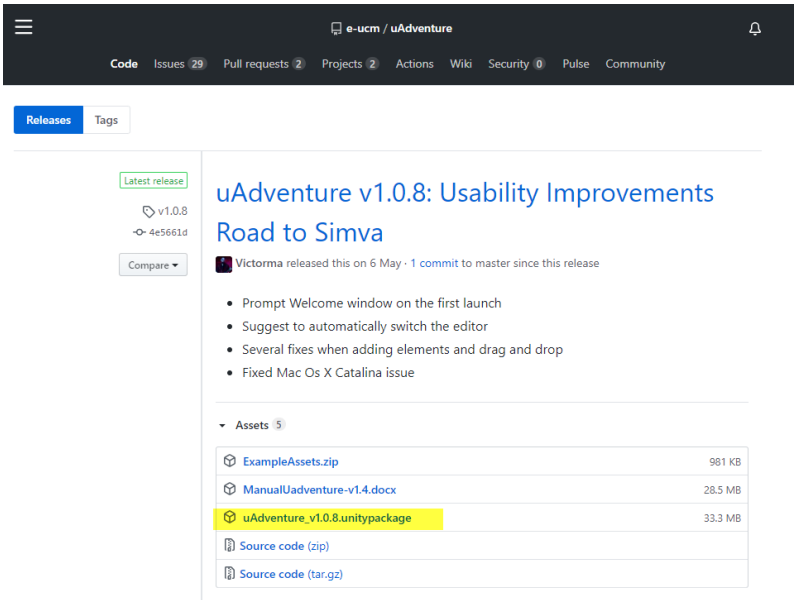


Figure 6. uAdventure releases page.

[4] https://github.com/e-ucm/uAdventure/releases

http://u-Adventure.e-ucm.es

There are two ways to import the package. The first and easiest is just by double-clicking on the package file once our project is opened. This will open the "Import" dialog showing a file description of the package to be imported (Figure 7). The second method consists in using the Unity editor import option. This option can be found in the top contextual menu in: "Assets > Import Package > Custom Package…". Once clicked a file explorer to find the project. Finally, as the package is selected the package importer dialog will show it (Figure 7).

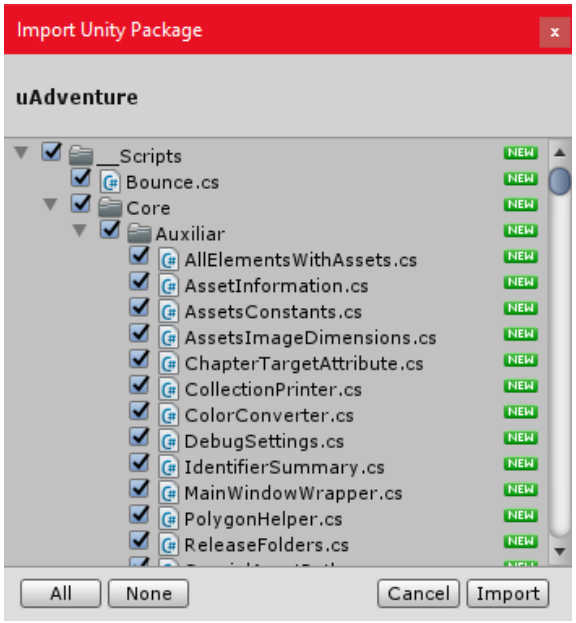


Figure 7. Unity package import window.

To finish the <u-Adventure> import, click in the "**Import**" button. The package starts its importation. There is a chance, if the Unity editor is in a different version than the <u-Adventure> recommended version that a pop-up will propose to upgrade the scripts. Just click on "**Go Ahead**, I've made a backup" to continue with the import.

If everything is correct, you will be able to see the uAdventure section in the top contextual menu (Figure 8).

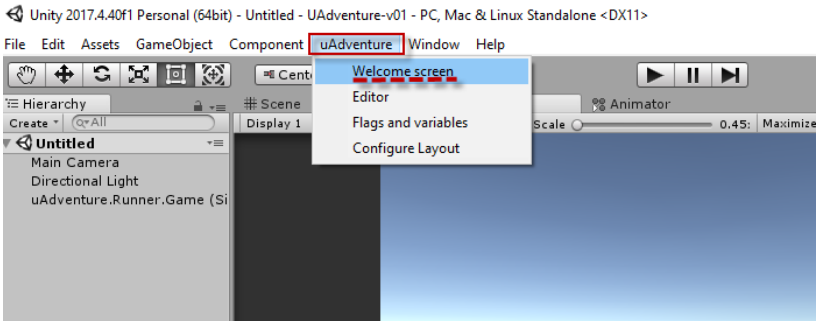


Figure 8. uAdventure Contextual Menu

If you are not able to see the menu, please check that the Unity project is using .NET 4.X before opening an issue. To check it go to "Edit -> Project Settings" and enter the "Player" section. In the right part of the screen, scroll down to "Other Settings" and find the "Configuration" subsection. **The value of "Api compatibility level" must be ".NET 4.X".**

1.3.3.2 Installing through the Unity Assets Store

Installing from the Unity Assets Store is not available yet. Sorry!

http://u-Adventure.e-ucm.es

1.3.4. uAdventure Project folder structure

Unity projects are defined by a folder containing an “Assets” folder and a “Project Settings” folder. In most projects Unity will generate another folders such as “Library” or “Temp” that can be regenerated. Other common folders are “Packages”, “Builds” or “Logs”.

uAdventure is installed inside of the main Unity project “Assets” folder in several folders: “uAdventure”, for the main functionalities and the uAdventure game data; “uAdventureAnalytics”, including learning analytics features (see section 6); “uAdventureGeo” for the new GPS and Augmented Map features (see section 4); “uAdventureQR” for the new QR code scanning based features (see section 5) and “uAdventureUnity” including experimental integration with native Unity scenes.

Once uAdventure is installed, it will manage its resources and data inside of the “Assets/uAdventure/Resources/CurrentGame” folder. This folder contains a classic e-Adventure project structure, including both “descriptor.xml” and “chapter1.xml” files that contain all the game model and a sub folder named “assets” where all the game assets will be contained including backgrounds, item images, character images, animations, etc. This “assets” folder should not be misled with the main “Assets” folder in the Unity project.

1.4. Starting out with the editor: basic project management and running games

In a <u-Adventure> Unity project we can open three different windows by using the uAdventure section in the top contextual menu. First the “uAdventure welcome window” (Figure 9) allows for configuring and creating the basic folder structure for the assets in your project, depending on the selected game type. Second, the “uAdventure editor” allows to manage the current project. Other windows will be explained in further sections. First, we will start configuring the game type in the welcome window.

1.4.1. Configure the <u-Adventure> project

In the “uAdventure welcome window” it is possible to regenerate the project folder structure and set up the game type. Even so, this step is not explicitly needed as the uAdventure package includes the default folder structure.

Two different game types can be selected, each represented by its own icon:

Third person games: These games, like the *MonkeyIsland*™ saga, have a visible player representation, shown as an avatar in the game which the player controls. The movement of the avatar requires time (the avatar must walk from one place to another) and the text is shown directly above its head (just like in a comic book).

First person games: The player has no avatar, thus the exploration in the game is in first person and with instant consequences for the actions. The *Myst*™ saga is an example of this type of games. This sort of game is usually better suited for photo-realistic and simulation games, in which photographs are used as the scenarios of the game.

http://u-Adventure.e-ucm.es

Figure 9. uAdventure welcome window.

Once clicking on the "New" button, files well be re-created and the game type configured.

1.4.2. Configuring the Unity layout for uAdventure

uAdventure is implemented in Unity and due to this reason, it comes with multiple extra functionalities coming from Unity engine. For this reason, to reduce the visual complexity and provide a simple and uAdventure focused experience there is a uAdventure layout option (Figure 10). By choosing this option, the default Unity editor will close all the Unity windows (including the scene editor, the inspector and the hierarchy among others) and will open the uAdventure main editor on the center and the variables and flags window on the right.

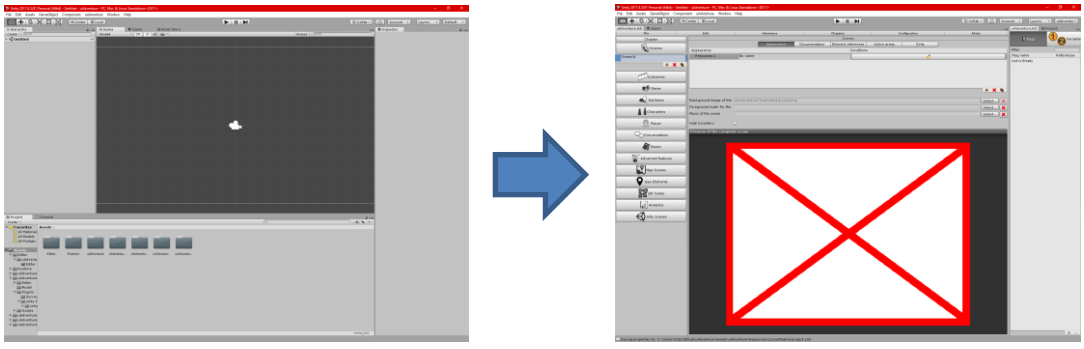


Figure 10. On the left, Unity default Layout. On the right uAdventure Layout.

Once a Unity project with uAdventure is opened, the uAdventure menu will appear on the top menu bar. We strongly recommend previous <e-Adventure> users to use the pre-configured uAdventure layout to simplify the uAdventure usage, avoiding the Unity windows. To use the uAdventure layout just click in the top menu in "**uAdventure > Configure Layout**". The Unity

http://u-Adventure.e-ucm.es

editor will restart and show the uAdventure main editor. Otherwise, you can just open the uAdventure editor window at "uAdventure > Open uAdventure Editor".

1.4.3. Importing a previous <e-Adventure> project

To import a project previously created on <e-Adventure>, you can use the "Open" button on the top of Figure 3 which allows for importing older <e-Adventure> projects in the current Unity <u-Adventure> project.

The open dialog allows for the selection of an *eap* file (or a project folder) to be opened. Once the file is selected, clicking the "open" button will start the import process and the editor will show the selected game project. Notice that by opening a project all the previous project data is removed, including both model and assets.

1.4.4. Building a project

Creating an executable file for our project is the last step in our game creation. The result of the build is going to be a native set of files for the targeted platform. This set of files will be different depending on each platform. Each build done using uAdventure is a simplification of the build pipeline of the Unity editor to make the process easier for non-experts. Later in this section we will explain how to use the native pipeline to build a uAdventure project.

The simplified builder can be found in the "File" section of the "uAdventure Editor Window". Several different options for each platform are available and will directly build the game unless some configurations are missing. By selecting "File > Build project… > Build project…" you can access the simplified builder configuration window, where you can select the different platforms to build and configure elements such as the game name, the author, the path among other things. The platforms available by default are Windows, Linux, Mac OS X, Android, iOS and WebGL. There are some characteristics and dependencies for each of them:

- Windows: The result of a windows build is a set of files including an executable "**.exe**" file, a "UnityPlayer.dll" and a folder with the "_Data" suffix containing all the resources and dependencies of the game. The Windows build does not have any external dependencies. More information can be found in the Unity manuals[5].
- Linux: The result of a Linux build is a set of executable files ".**x86**" and ".**x86_64**" that will run in x86 and x64 platforms respectively; and a "_Data" folder containing all the resources and dependencies. The Linux build process does not have any external dependencies.
- Mac OS X: The result of a Mac build is an "**.app**" folder.
- Android: The result of an Android build is a ".**apk**" file that can be installed in the targeted and higher Android platforms. To build for Android, an updated version of the Android SDK and Java 8 is needed. Also, a keystore and a password are needed to sign the application. Complete and updated guidelines of how to export in Android can be found in the Unity manuals[6].
- iOS: The result of an iOS build is a XCode project. In contrast to all the other builds, it requires another step to generate the executable file. The requirements for this build are: a computer running Mac OS X with XCode installed and updated and an Apple Developer License (the license is not required for debug builds). Complete and updated

[5] https://docs.unity3d.com/Manual/WindowsStandaloneBinaries.html

[6] https://unity3d.com/learn/tutorials/topics/mobile-touch/building-your-unity-game-android-device-testing

http://u-Adventure.e-ucm.es

guidelines of how to finish this process can be found in the Unity manuals[7].

- WebGL: The result of a WebGL build is a folder with a "**.html**" file and its resources. These files are index.html and three folders (Build, StreamingAssets and TemplateData) and **must be put together in a web server for the project to work**. The index.html is the default Unity WebGL template and so the TemplateData folder is used only to display the different icons and images in this file. The Build folder contains all the Unity engine and game resources for the game and therefore this folder is required to run the game. Finally, the StreamingAssets folder contains all the videos used in the game and so, if there are no videos in the game, it might not be present.

Note that also, to be able to export the project in the selected platforms, the build platform must have been installed during the Unity editor installation. There are two ways to install support for new platforms. We recommend using the Unity Hub (Figure 11). In the Unity Hub you have to click on the "Installs" section on the left and click in the "⋮" button of the version you want to add the modules in. Then click on "Add Modules" and follow the wizard.

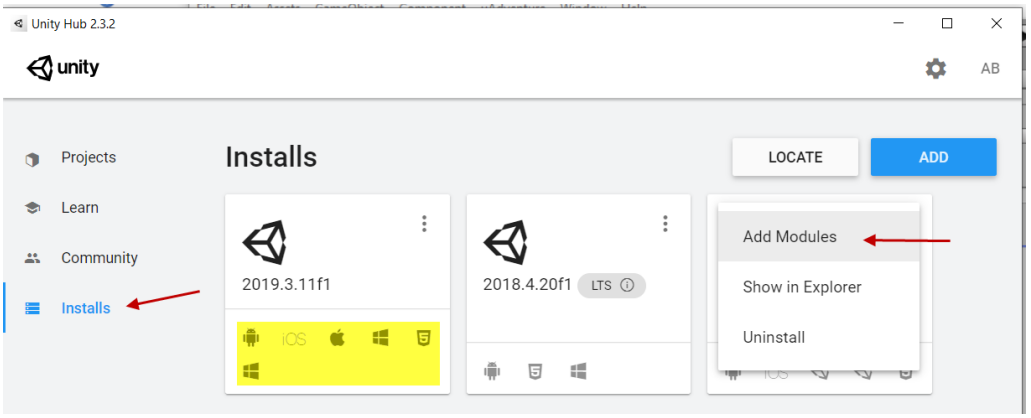


Figure 11. Adding modules to Unity from Unity Hub.

If you want to install a new build platform without Unity Hub you can also use the Unity editor and go to "File > Build settings…" in the Unity top contextual menu select the platform and follow the indicated steps to download and install the package. Interaction in <u-Adventure> games.

Although this document is oriented to explain the features of the <u-Adventure> game editor, it is important that potential users of the platform get an idea of what the games will look like when the engine runs them. For this purpose, it is important to know how interaction is carried out, which is explained in the next section.

1.5. Interaction in <u-Adventure> games

When the game is launched, the player will perceive the scene as it is, without further menus or head-ups displayed. Players can explore the scene with the mouse. When the mouse pointer is over any interactive element, the cursor will change and, if configured, a brief text may also be displayed.

The player can interact with different elements (e.g. characters, items, exits) by clicking the left button of the mouse (this differs from previous <e-Adventure> games where interaction with items was performed with right-click). This change aims to ease playing in mobile platforms, where the only way to interact is touching the elements that appear on the screen. When interacting with an element, a contextual menu with different interaction options will appear (Figure 12). Press on any of the options of the menu to trigger the action.

[7] https://unity3d.com/learn/tutorials/topics/mobile-touch/building-your-unity-game-ios-device-testing

http://u-Adventure.e-ucm.es

Figure 12. Example of a contextual interaction menu in an <u-Adventure> game.

There is a structure called **"Inventory"** where all the objects collected by the player during the game will be stored. This inventory can be accessed by moving the mouse cursor towards the bottom or the top of the scene, although this behavior is configurable.

To display the game menu, press the 'Esc' key.

1.6. Run menu

This menu allows running the games from the <u-Adventure> editor, which is very convenient to test while the game is being edited. To do this just use the play button in the middle top of the Unity screen (Figure 13). Please make sure you save your project either by clicking "File -> Save project" (Figure 14) or using the popup that will appear when pressing the play button. Automatically, the editor window will be swapped with the game window.

Figure 13. Unity run controls.

Notice that when using the run menu from Unity the execution is going to be slower than when executing a native version.

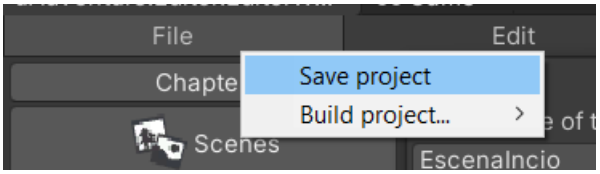


Figure 14. Save project.

2. My first <u-Adventure> game

We will start by studying the options available in the editor (Figure 15). In the left side, we find a structure made up of collapsible panels (the structure panel) used to organize all the elements in the game. When an element is selected in the structure panel, the properties of that element will be editable in the main part of the screen (the right-side panel). This way, the structure panel is used to access the different elements, modify them, delete them and create new ones. To perform some of these operations, a common set of buttons is used to: add new items (), delete items () and duplicate items). These same icons are used in other contexts in the editor. A first Chapter and Scene are created by default.

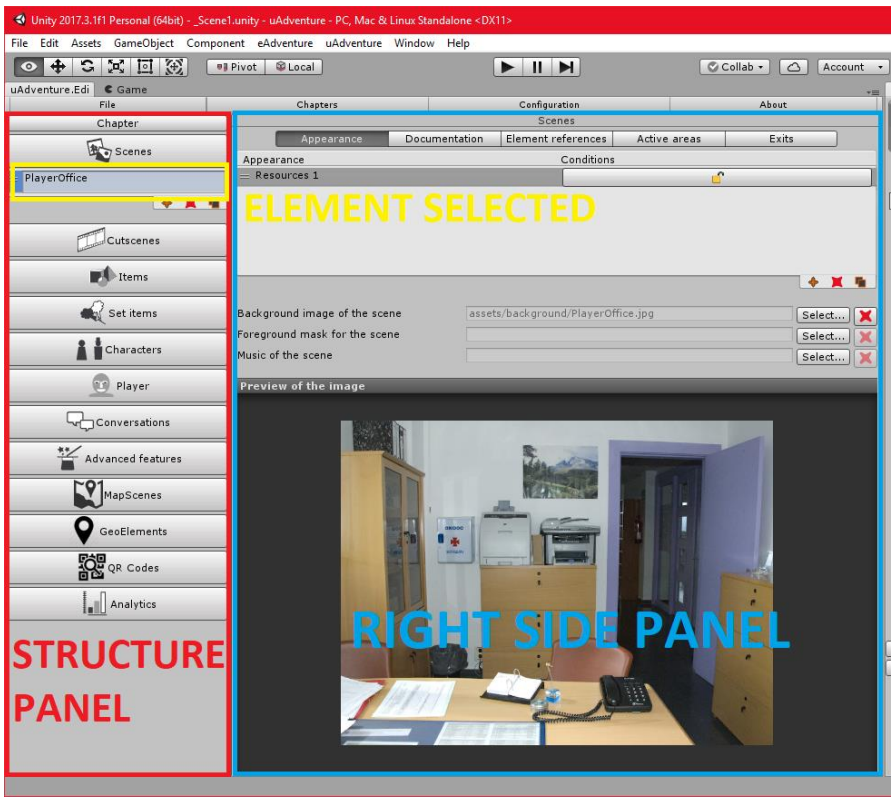


Figure 15. Initial view of the <u-Adventure> editor, once a new project is created or an existing one opened.

Now we will proceed to study each element in the structure of an <u-Adventure> game while we create one sample game. Our game will be called “FIRE PROTOCOL”:

In a pleasant summer morning at the campus of the Complutense University of Madrid, a fire alarm has gone off in Office 411. You are in your office and receive a phone call with instructions to evacuate the building and making sure that everyone in that floor leaves the building. The fire protocol to be followed has some specific recommendations that will warranty the safety of everyone. If you do not follow the instructions correctly, someone could die.

In this story, there will be one boy in danger (besides you). Your mission is to get him out of the building without risking your life or his.

2.1. Chapters

<u-Adventure> games can be divided into chapters. This allows games to be fragmented into smaller parts that are easier to manage, design, edit and maintain. Each chapter can be viewed as a totally independent mini-game, or as an act in a theatre play. Elements on a chapter are only available within the chapter and not from other chapters in the game.

The editor can only edit one chapter at a time. The "Chapter" menu (found in the structure panel) has options to create, import, delete and organize the chapters in the game, as well as to modify the current chapter's name. This menu includes options to adjust the *flags* and *variables* available in the chapter; however, notice that the chapter flags and variables menu will appear on the right side of the main view, as detailed later.

Once a chapter is selected, the title, the description and the first (or initial) scene of the chapter can be modified (Figure 16). First, you can define the title for this first chapter which will be called *Save Balta*. You can optionally give a description of the chapter. The default scene will belong by default to this chapter.

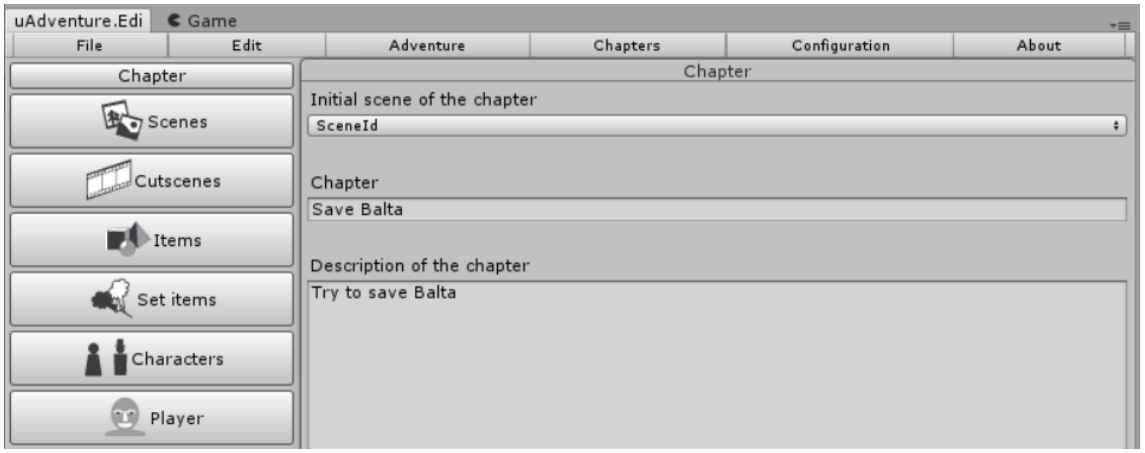


Figure 16. Chapter edition including name, description and initial scene.

2.2. Scenes and cut-scenes

<u-Adventure> chapters are organized into *scenes* and *cut-scenes*. On the one hand, *scenes* are the scenarios or places where the action of the game takes place (Figure 17), that is, where the player interacts with the objects and characters in the game. Scenes are connected to other scenes by *exits*, as we will later explain. On the other hand, *cut-scenes* can be used to increase the educational value of games. Cut-scenes can be either *Slidescenes* or *Videoscenes*. *Slidescenes* are made of slides or images that are shown using the full screen. *Videoscenes* will also display a video using the full screen.

2.2.1. Adding a new scene

To add a new scene, the *scene* or *cut-scene* element in the structure panel must be selected, depending on the kind of scene that is to be added. When the kind of element is selected, the list with all the elements of that kind is displayed. Besides, a button to add a new element (✚) appears to the right. Clicking on the button adds a new element with a default identifier (from now on, id). **The ID can be modified by selecting the element, which will change its appearance to a text field, and must be unique. Ids must contain letters and numbers only and start with a letter. This applies to all ids in <u-Adventure>.**

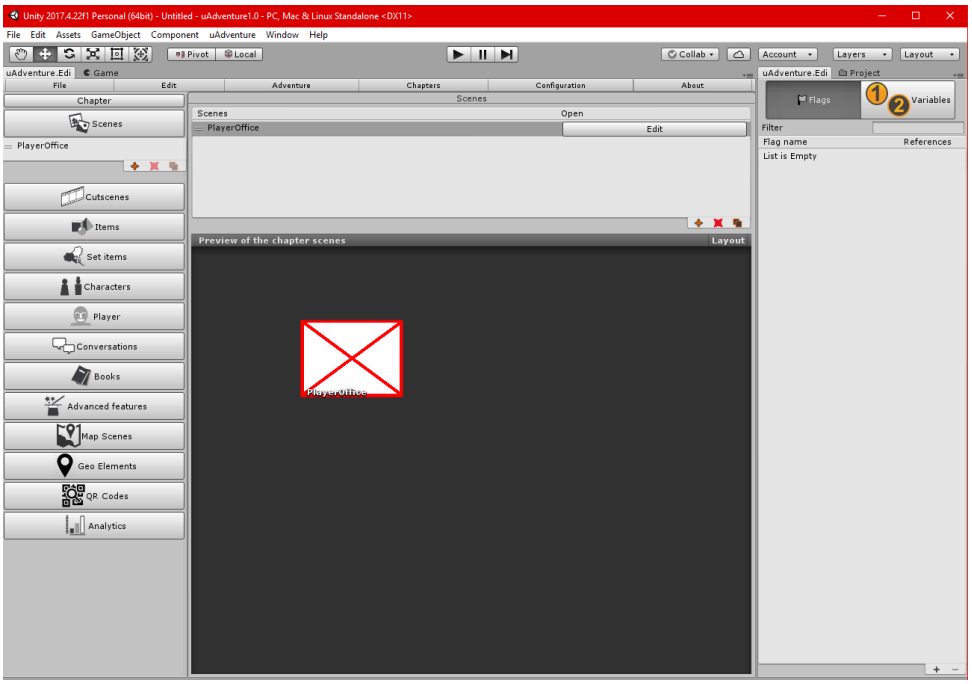


Figure 17. Scene edition shows a preview of all the scenes in the chapter. By default, games include an empty scene.

When a scene is selected (Figure 18) in the structure panel, five tabs appear in the right-side panel (if the game is a third person game, seven tabs will appear). The “Information” tab will appear first and this is common to most <u-Adventure> elements. In this case it allows for the edition of the scene general description and its name. **The element name should not be confused with its id; the first one is used in the game while the second is unique and shown only in the structure of the game.** The name of the element can have white-spaces and use any character.

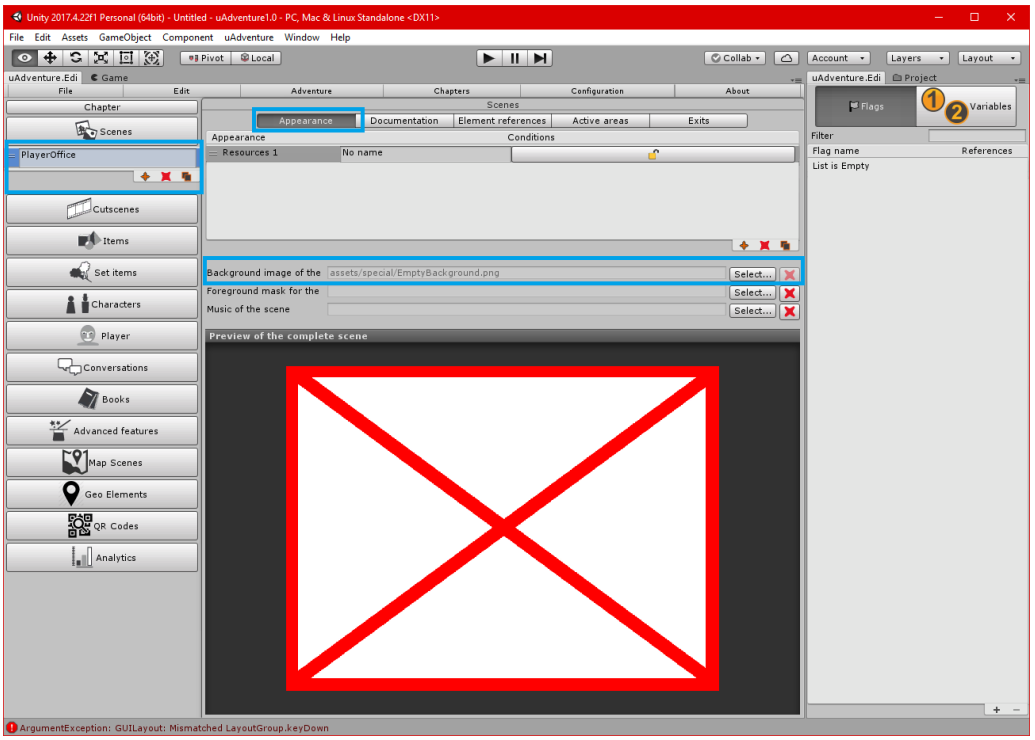


Figure 18. Scene edition panel, with the "Appearance" tab selected.

The "Appearance" tab is used to change how the scene is shown (or how it is perceived). This applies for any other element in <u-Adventure> that has an "Appearance" tab. For the scene appearance, three resources or assets can be edited but only the background image is required while the other two are optional. The contents and meanings of the other tabs will be studied later in this guide.

2.2.1.1 Background image (required)

This is the most important asset of the scene and represents the actual scene in the game. The background images must be at least 800x600 pixels in size. However, although all scenes must be exactly 600 pixels in height, the width of the scene can be bigger. This way, in third person games, the scene will move along with the player's avatar when it reaches the border of the screen. In first person adventures, a cursor (button) will appear to each side of the screen allowing the user to move the background as needed during the game. **Images for the background, as any other assets in <u-Adventure> can be: PNG, JPEG (or JPG) or BMP.**

2.2.1.2 Foreground mask (optional)

The foreground mask (Figure 19) is a black and white image that identifies which parts of the background image should be drawn in front of the objects, characters (NPCs) and the player, and which parts should be drawn behind.

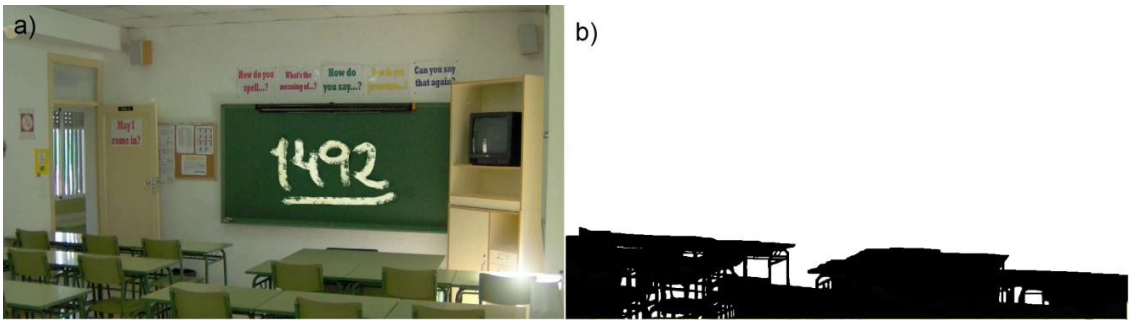


Figure 19. Background image (a) and foreground mask (b). The black area of the mask determines the parts of the background image that will be painted in front of other elements while the white area determines those that will be painted behind.

Foreground masks are obsolete and their use is not encouraged, as they are rather complex to use and present many limitations. This behavior can be reproduced using *set items* which provide more flexibility and are explained later in this guide.

2.2.1.3 Background music (optional)

This is a music track that will play in a loop while the scene is on screen. It supports many formats as: mp3, ogg, wav, aiff, mod, it, s3m and xm. Full list can be found at the Unity manuals[8].

2.2.2. EXAMPLE: Defining the assets of a scene

To start with, we can modify the attributes of the default scene in the "Appearance" tab. In this example, our player will start this adventure in the reception, so we will call this Scene *PlayerOffice*. We will add an image as background of this scene which will be *PlayerOffice.jpg*[9]. When a resource is added (the background for instance), it is copied into the *assets/background* folder of the project.

To add a background to a new scene, we start out by clicking the "Select" button next to the

[8] https://docs.unity3d.com/Manual/AudioFiles.html

[9] All the assets required in the examples can be downloaded from the uAdventure releases sections together with the UnityPackage and the Manual.

asset. Clicking this will display a dialog to select an image (Figure 20).

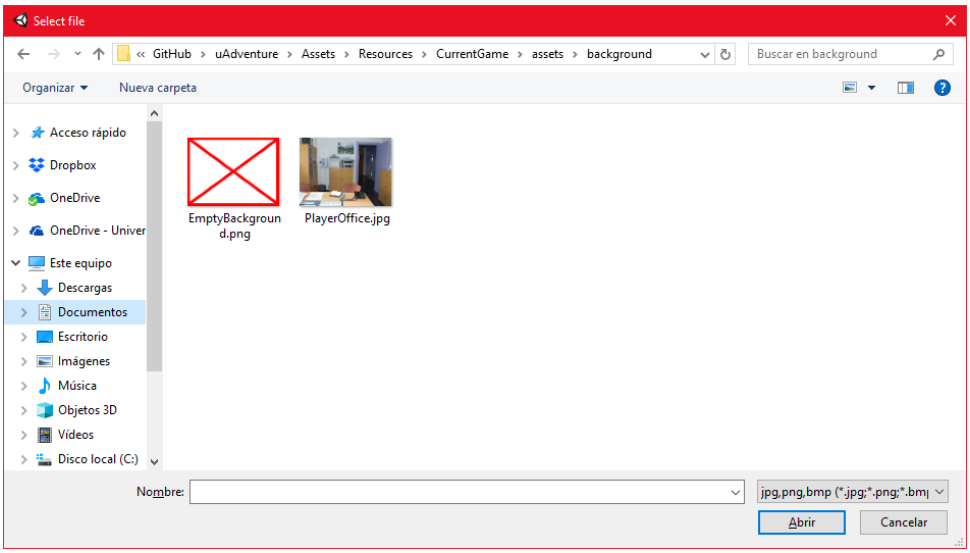


Figure 20. Assets selection dialog for the scene background.

Dialogs, like the previous one, are used to select a valid resource for the file system (in this case, an image). Notice that background images must be at least 800x600 as previously established. By clicking the "Open" button, the image asset will be automatically added (copied) to the project and configured to be compatible with Unity importer (enabling the Read/Write property).

The dialog shows by default the project folder so the files used in the project can be easily selected from the different project sub folders (i.e. backgrounds folder "*name-v01\Assets\uAdventure\Resources\CurrentGame\assets\background*").

In this case we will not use a foreground mask because there are not going to be any elements in between, for example, behind the desk. However, to configure it a grayscale image has to be used to determine which layer is always on top.

After these steps are completed we can see the result in Figure 21.

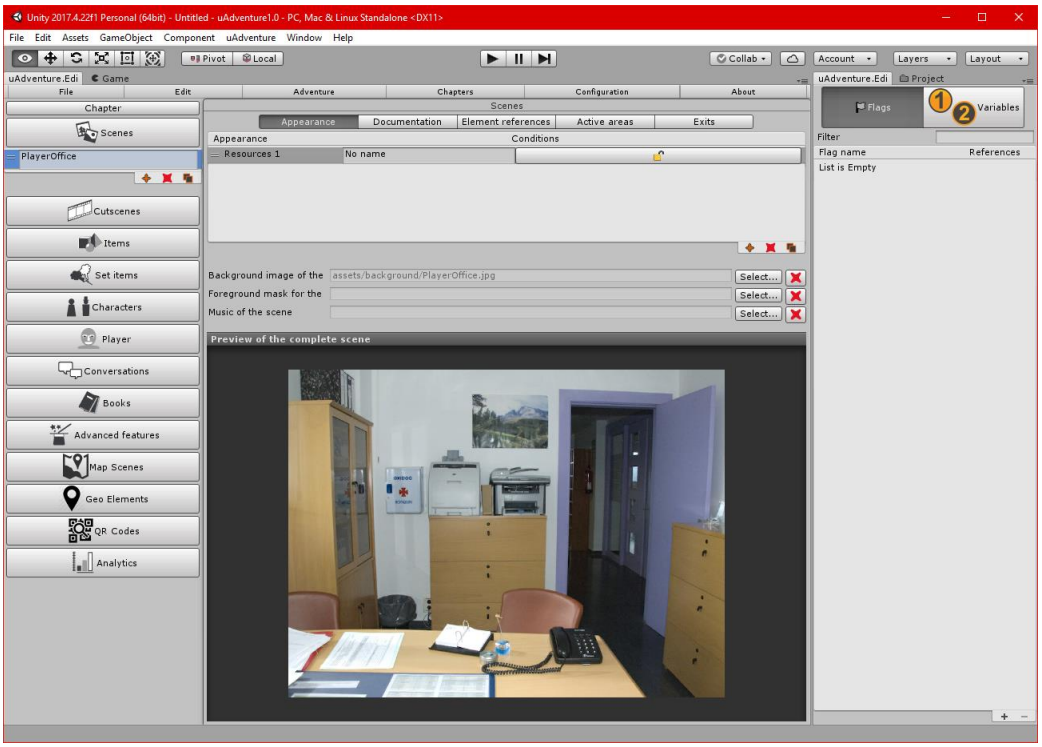


Figure 21. Resulting *PlayerOffice* Scene.

Before the previously modified scene is shown to the player, it is possible to add a cut-scene that will introduce the game. In this case, the cut-scene will be the starting point of the adventure.

2.2.3. Connecting scenes: adding an exit

If we want to connect different scenes, we can add exits to the scenes in the game. Without exits, the game would mostly take place in the same scene. Exits can be shaped as invisible rectangles or polygons. When the player clicks on an exit, the scene changes (in the case of third person games, only after the avatar reaches it).

2.2.4. EXAMPLE: Creating an exit

To better understand this concept, suppose we have two scenes: the one we created in the last example (called “PlayerOffice”) and a new one called “DepartmentArea”. We can use the open door in the office shown in Figure 21 to simulate an exit in that area that will allow the player go from the office to the department area.

The first step is to create the scene “DepartmentArea” with its own background (Figure 22).

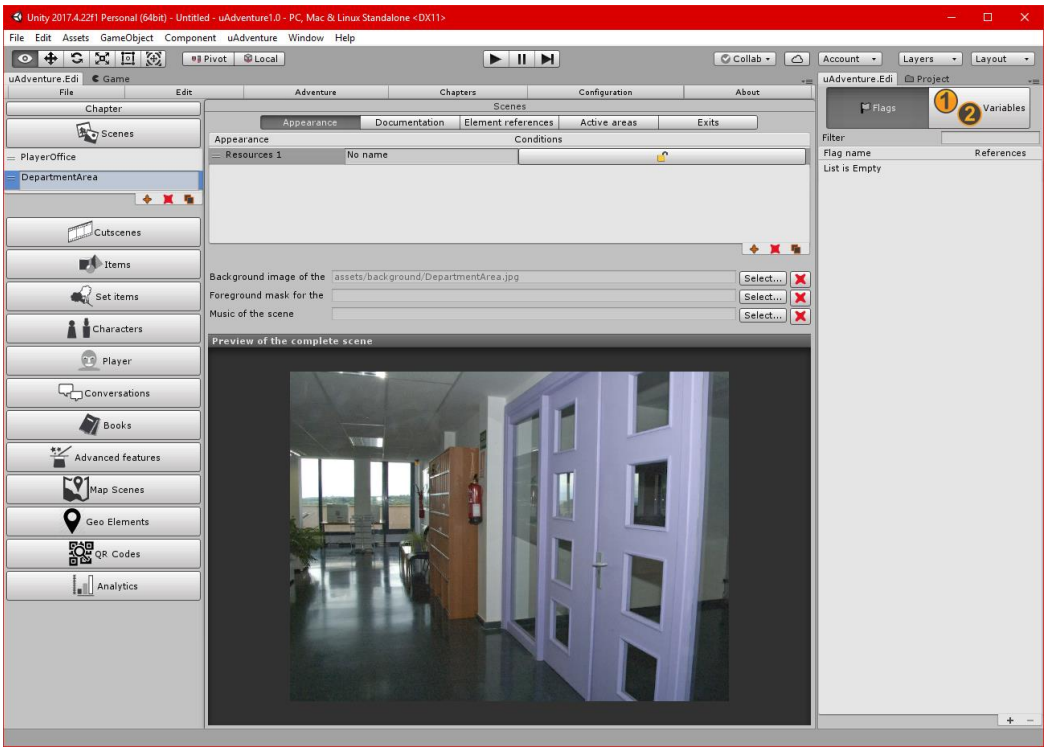


Figure 22. Department Area scene.

The next step is to select the "Exits" tab for the "PlayerOffice" scene. This shows in the right-side panel a list of exists and a preview of the scene. At the right of the table, a (✚) button can be used to add a new exit. When clicking on it, we have to select the scene exit target from a drop-down list of the available scenes. After adding the exit, a new red rectangle will appear in the scene and the corresponding new line will be added in the top list of exits (Figure 23). To change the destination, click on the element in the list or in the scene and select the destination in the list.

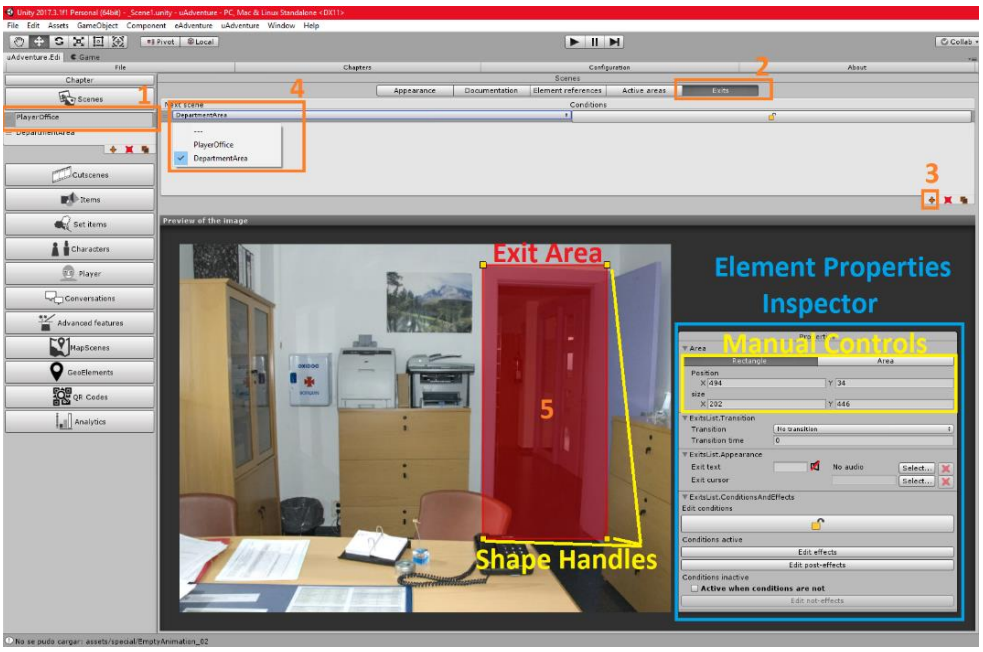


Figure 23. Creation and configuration of an exit. Steps are 1), select the scene, 2) select the exit

tab, 3) press the + button, 4) select the element and pick a destination and 5) define the area.

Now that we have created the *Exit*, we are going to define the area and position. The idea is to cover all the space inside the door frame, so the player will be able to use the Exit by clicking anywhere inside the area of the frame. To change its shape and properties, select the exit by clicking on it and change the shape using the yellow shape handles or the manual controls at the right "Element Properties Inspector" (Figure 24).

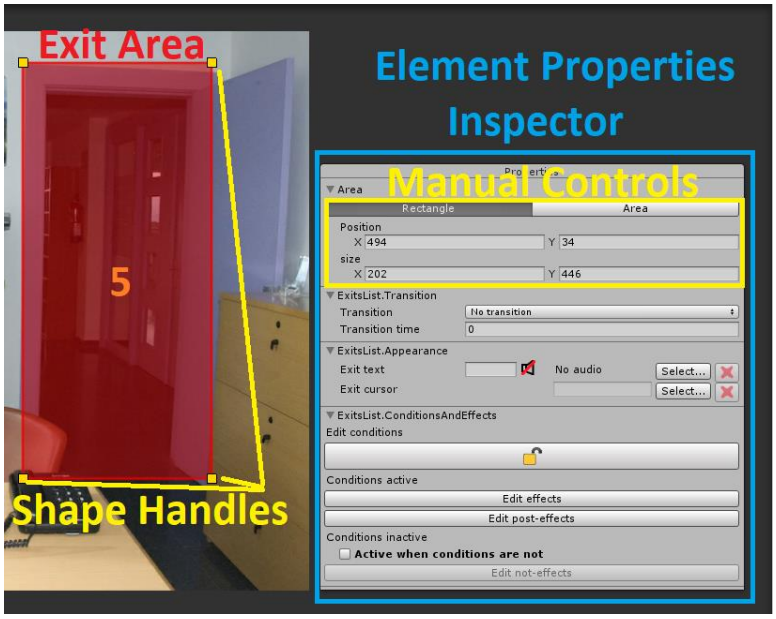


Figure 24. Element properties inspector and area shape.

Optionally, you could change the shape from "Rectangle" to "Area" making it possible to define any polygon shape. This "Area" mode is controlled by three tools: move, add and remove (Figure 25). In the "Area" mode, the handlers become blue and round to easily identify the mode.

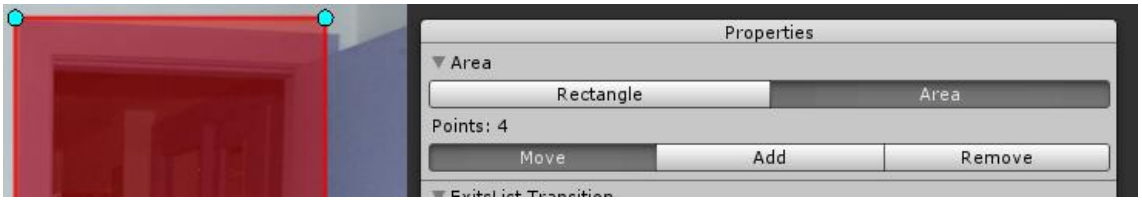


Figure 25. Area shape mode showing the different control tools and the blue roundly handlers.

When the exits are configured, a preview of the different scene navigations can be seen in the main scenes panel that is access by clicking the "Scenes" left box (Figure 26). This view can be used to determine if the scenes are well connected and give insights of how the game scenario is organized. The button "Layout" on the top-right part of the view automatically organizes the scenes on the view.

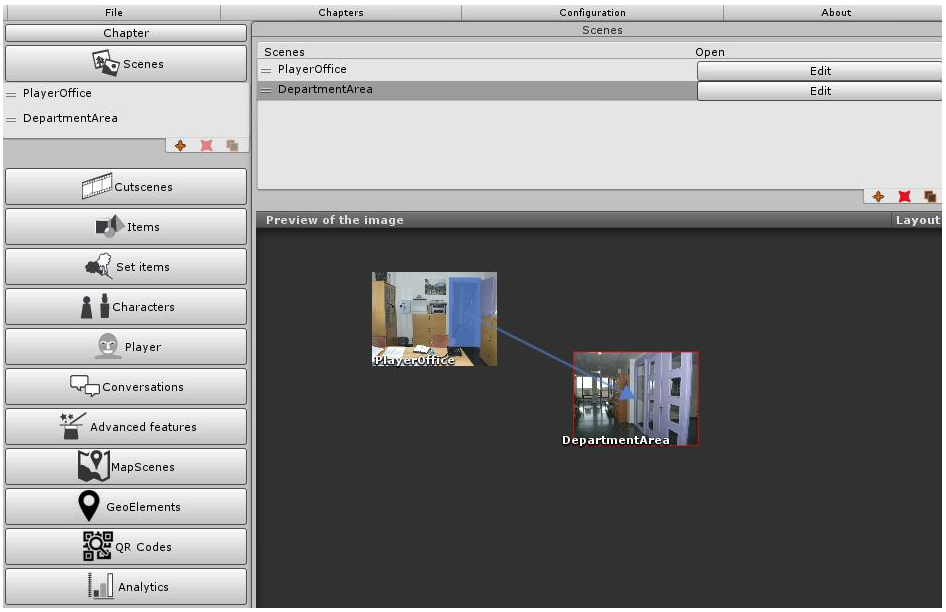


Figure 26. Main scenes preview displaying two scenes connected with an exit.

Now you can run your game again and see how you can move from the first scene to the second one by clicking in the "invisible" area that is located on the door frame (Figure 27).

Figure 27. Expected exit behavior.

Now, our game will be about a Fire Protocol that must be followed to save someone from the fire. The next step is to add an item to interact with in our office scene.

2.2.5. Scene initial position

In third person games only, where the player is visible in the scene, before being able to test the game, the player assets must be defined; this is studied in detail in section 2.9. However, it is possible that the player appears in an apparently random position. This can be changed by defining an initial position, which we will do for the office scene. To do this, choose the "Player movement" tab while the "Office" scene is selected in the structure panel.

This tab will show a preview of the scene with every element defined in it in grey (Figure 28). In this case, the "Use initial position" choice is selected. The player can be dragged to the desired position in the scene. The player scale in the scene can also be modified by dragging the corners of the element.

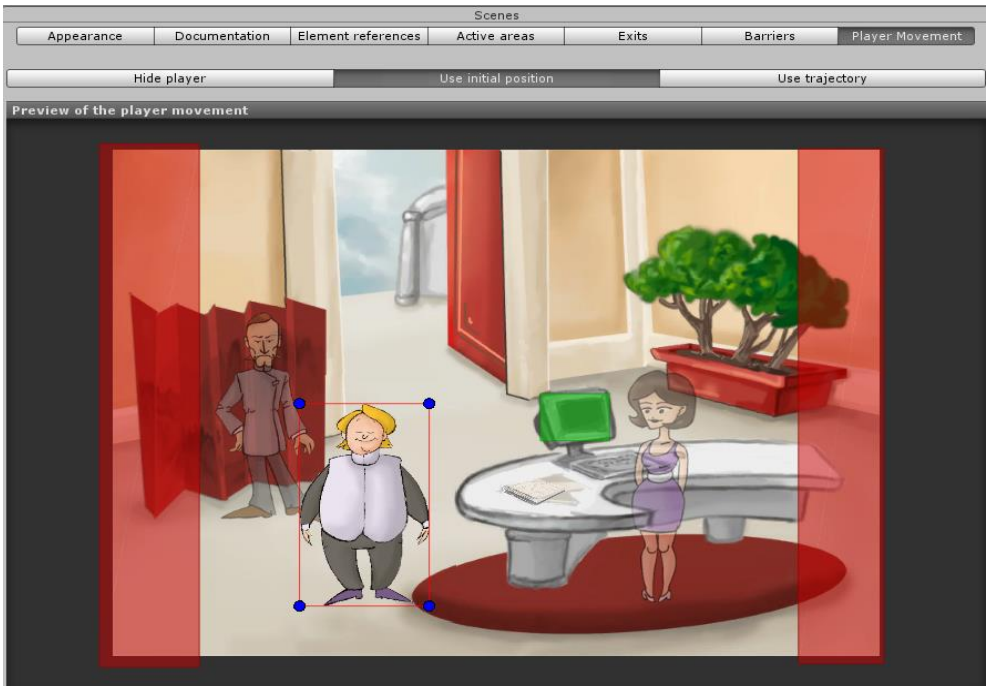


Figure 28. Player movement tab with use initial position enabled.

2.3. Cut-scenes

<u-Adventure> cut-scenes are used to display multimedia content that are shown in full-screen mode, have limited user intervention and after finishing are followed by new scenes or cut-scenes. There are two basic types: Slide-scenes and Video-scenes. Slide-scenes are made up of a series of images. Although Video-scenes can use videos in many formats, we suggest using OGV, VP8 and WEBM, as those formats are the formats compatible with more platforms. Full list of video format support can be found at the manuals[10].

2.3.1. Slide-scenes

To add a new slide-scene, select the Cut-scene left menu and click the add button. When clicked, two options appear for slide-scenes and video-scenes (Figure 29). Choose "Add slide-scene" to continue. Selecting the slide-scene will result in a new panel with three tabs in the right (Figure 30).

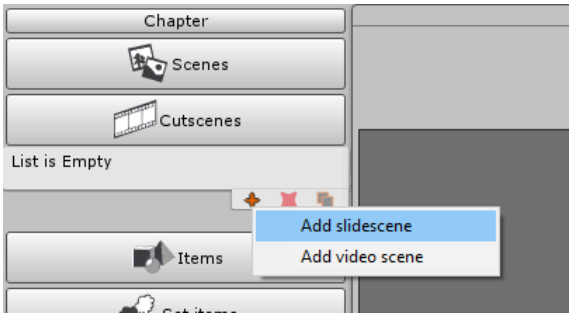


Figure 29. Create cut-scene options.

The three tabs in the slide-scene edition panel are:

- "Appearance": this is used to set up the images that make up the slides of the cut-

[10] https://docs.unity3d.com/Manual/VideoSources-FileCompatibility.html

scene. Slidescenes use animations, which can be configured by clicking the "create/edit" button and using the animation editor (see section 3.8.1). *Additionally, a deprecated method can be used to load animations made of consecutive files ended with "_01", "_02", "_03", etc. for each frame but animations are highly recommended as times and transitions can be configured in between.*

- "Documentation": as in other <u-Adventure> elements, the documentation is not used by the game and just provides additional information for developers.
- "Cut-scene end configuration": this tab is used to establish what happens when the cut-scene ends, see 2.3.3 for details.

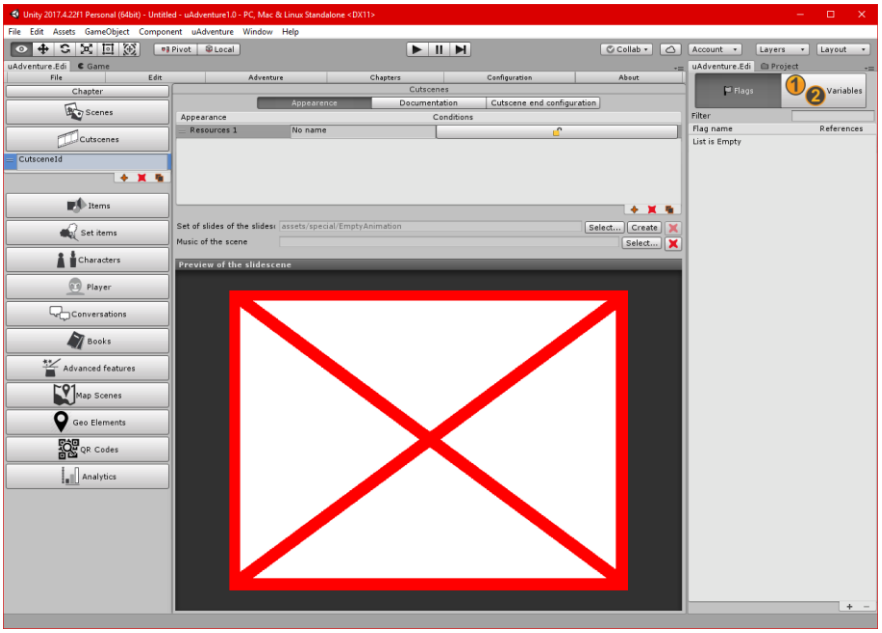


Figure 30. Slidescenes tabs.

2.3.2. Video-scenes

Video-scenes are added in a similar way to slide-scenes. The tabs in the video-scene edition panel are the same that are found in the slide-scene panel, with the only difference that the asset required in this case is a video file. After selecting a file, a "Play" button allows us to test the video and its compatibility with the platform.

<u-Adventure> supports some kinds of MOV, MPG and AVI files. MPG files must use the MPEG1 video codec. MOV and AVI files can use DIVX4, XVID, DIVX3low, S-Mpeg 4 v2 and DIVX5, although further restrictions apply.

2.3.3. Connecting cut-scenes with other scenes

Just as regular scenes, cut-scenes can be connected to other scenes in the game. The process is similar to the one used for regular scenes; the difference is that cut-scenes only have one exit. This is configured in the "Cut-scene end configuration" by choosing one of the available options (see example later on Figure 34):

- Returns to the previous scene: if selected, when the cut-scene ends the game will go back to the previous scene (either regular or cut-scene). If the cut-scene is the first in the game, an error will arise, and the user will be informed.

- Goes to a new scene: This allows to choose a new scene to be triggered once the cut-scene ends. There are some details to be configured:
  - *"Next scene"*: The ID of the next scene in the chapter, to be selected from a combo box.
  - *"Edit effects"*: Allows for the configuration of effects that will be triggered after changing the scene to the new one.
  - *"Use a destination position for the player"* and *"Edit the destination position"*: this option activates the selection of a destination position for the player in the next scene, for games in third person, and the button is used to choose the position.
  - *"Transition"* and *"Transition time"*: A transition (and its duration) can be selected for the change to a next scene in the game.
- Chapter ends: When this option is selected, the chapter will end, and the game will continue to the following chapter or will finish if it does not exist.

2.3.4. EXAMPLE: Creating an introductory slide-scene

The purpose of a slide-scene is to dynamically present non-interactive information. In contrast to the normal playstyle, in a slide-scene (or a video-scene), the player cannot make decisions or use its character and inventory. Therefore, slide-scenes (and video-scenes) are perfect to introduce contents.

We are going to use a slide-scene to introduce our game to the player by using several pictures with different messages. Also, some pictures are going to be used to introduce the faculty building.

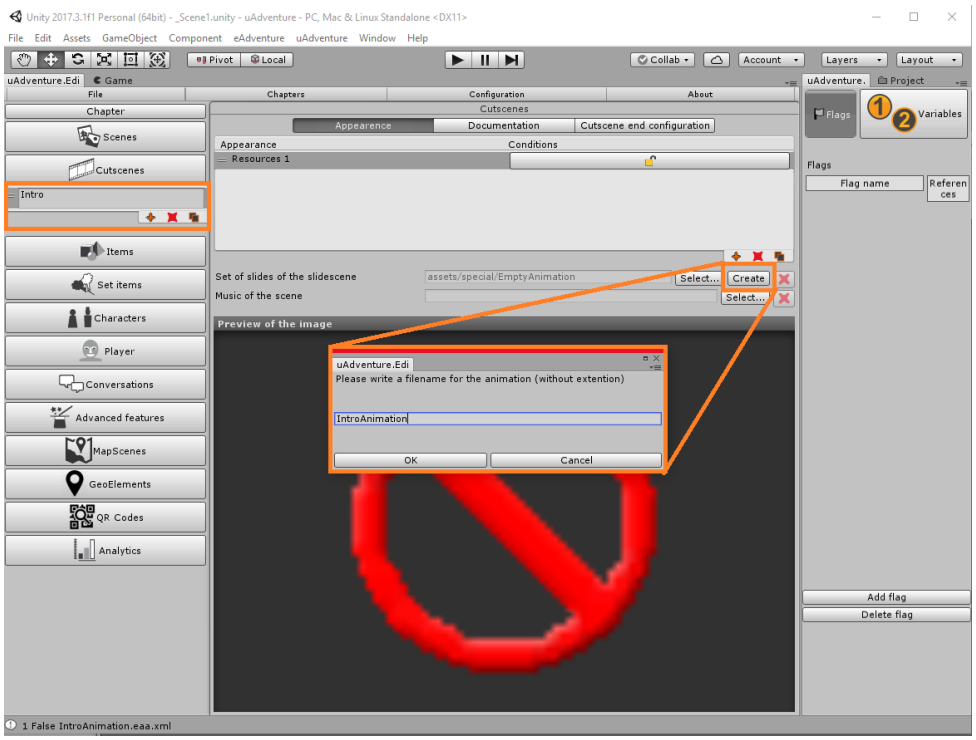


Figure 31. Slide-scene animation creation process.

We can start creating a new slide-scene and modifying the name of it to "Intro". Then we must select it and go to the "Appearance" tab, where we can see to see a big forbidden symbol. That symbol means that our slide-scene is still using the default empty animation. To create a new animation for our slide-scene we are going to click on the "Create" button in the "set of slides" field.

If we already had an animation to re-use, click in the "Select…" button. After clicking the "Create" button, a prompt asking for a name appears (Figure 31).

Right after the animation is created, the Animation Editor (Figure 32) is prompted.

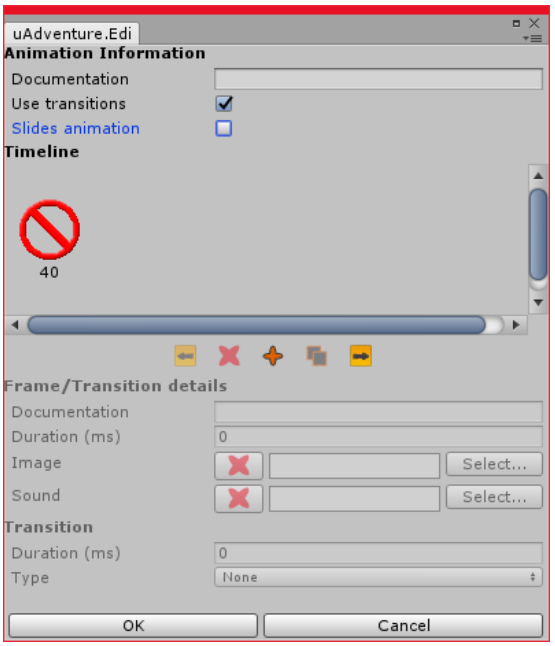


Figure 32. Animation Editor with a single empty frame.

This animation will consist of two images: *Intro_01.jpg* and *Intro_02.jpg*, the former will show the message "Protocol of action to face a fire alarm in the School of Informatics", the latter will show a picture of the building with the message: "A pleasant summer morning at School of Informatics…"

To select the first frame, click on the image and the fields below will be now enabled to edit. Select the image in the image field and then, click on the (✚) button to add a second frame. You will see a square between the two frames representing the selected transition. However, since this is going to be a simple cut-scene we are going to disable transitions in the top checkbox ("Use transitions"). When you have selected the image for the second frame (Figure 33) click on "OK" for the animation to be created.

Figure 33. Finished animation frames.

Now, we need to tell the cut-scene to go to the Scene "PlayerOffice" once it ends. To do that, switch to the "Cut-scene end configuration" tab and select "Goes to a new scene" where you can choose the "Next scene" and set the value to "PlayerOffice" (Figure 34).

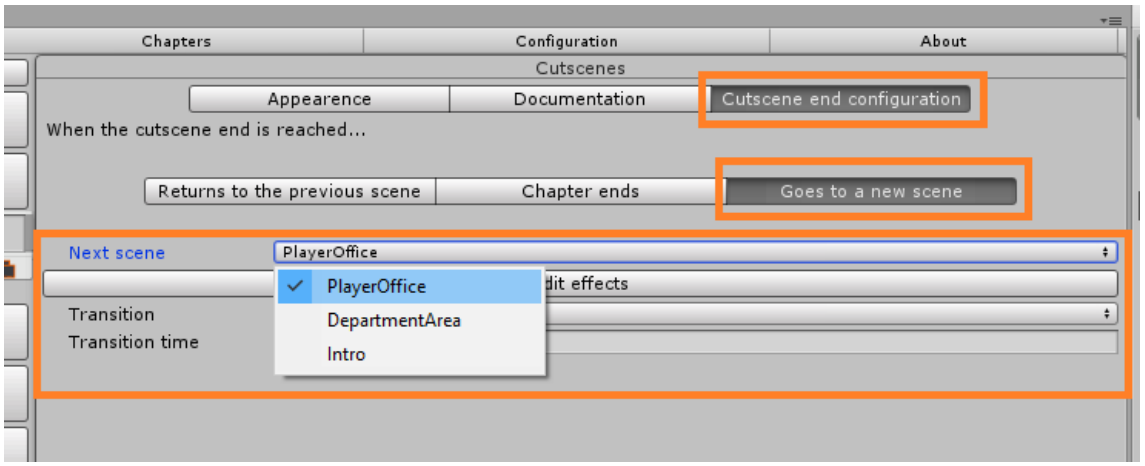


Figure 34. Cut-scene end configuration to go to a new scene.

Going back to where we started, we created this slide-scene to be the Intro of our game. This means that this scene must be shown as the initial scene, which is a special case of the chapter configuration. To change it, we are going to select the "Chapter" box in the left and change the "Initial scene of the chapter" value to our new "Intro" (Figure 35).

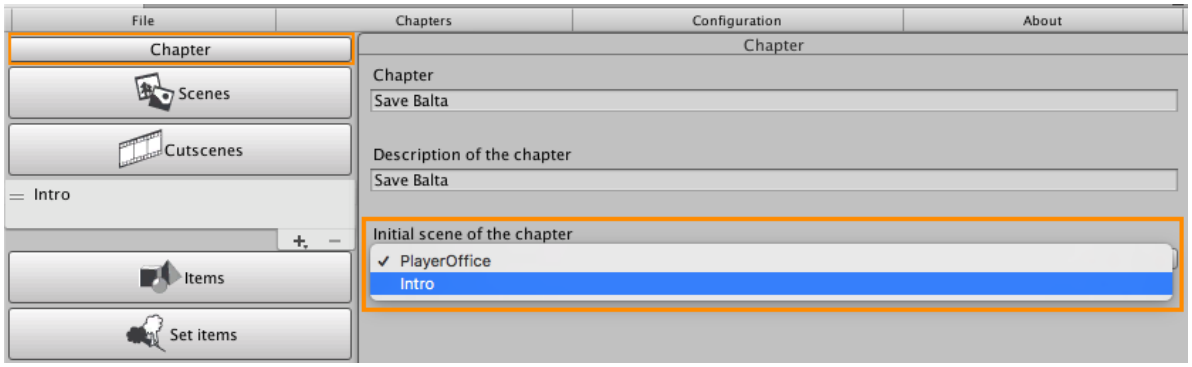


Figure 35. Chapter initial scene selection.

Now we can run our game to see the partial results.

So far, you have learned how to create a game with a Chapter, a Scene and a Cut-scene. You also know how to control the starting point of the game and how to change from a cut-scene to a scene.

2.4. Items

This section describes how to add new items (or objects) the player can interact with in <u-Adventure> games.

2.4.1. Adding a new item

Creating a new item is very similar to creating a new scene. It starts by selecting the "Items" element in the structure panel and click on the add button (). Click the button will add a new item with a unique default ID. This ID has the same properties, and is edited in the same way, as the ID for the scene as seen in section 2.2.1.

When the item is selected in the structure panel, the edition process is like the one followed for scenes (Figure 36). In the case of objects, the "Appearance" tab only requires that will be used when the item appears in the scene and an icon that represents the appearance of the item, will also appear in the inventory[11]. When an icon for an object is not used, the main item image is going to be shown as icon. In contrast to <e-Adventure>, the transparency of the images cannot be selected. However, files with transparency (in RGBA format) are supported and used straight ahead.

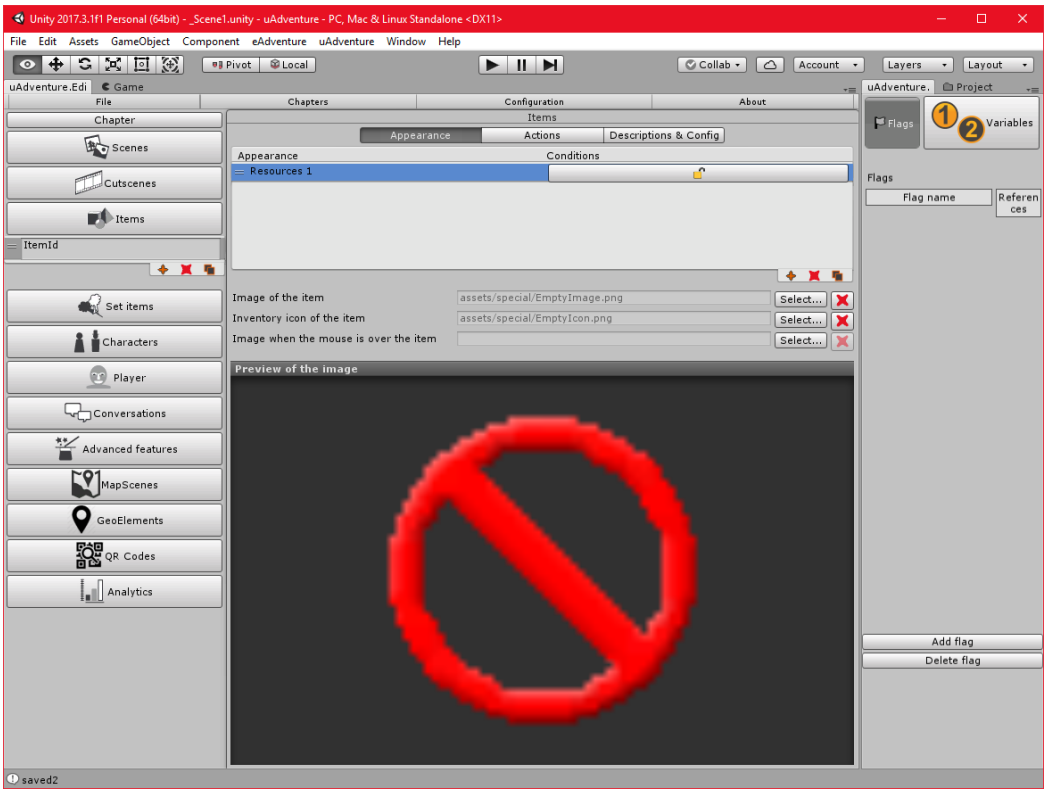


Figure 36. Items panel with appearance tab selected.

[11] As in traditional graphic adventure games, the player has an inventory where all items are placed once grabbed from the scene. These items can be used later in the game just as if they were in the scene.

2.4.2. Interacting with items: actions

Actions are the basic interaction system of a point and click game. In games like Monkey Island, the actions where shown in the bottom left corner of the screen and were selected by clicking on them. In uAdventure the interaction is simplified and defined in each game element. Both items, characters and active areas may contain actions that behave in very similar ways. Since actions are interactions, we must specify the consequences (the results) of each interaction (i.e. using a phone in the table). To specify them, an effect system is used and will be explained later in section 3.1 and more precisely in sections 3.1.9 and 3.1.10.

For items, if we select the “Actions” tab for an item, a list of actions will be displayed in the right-side panel (by default, it will be empty). This list will show five different columns: 1) action, where the type of action is identified; 2) description, where some information can be written just for documentation purposes; 3) needs to go, which indicates if the player has to be close to the object to do the action in third person games; 4) conditions, that enable or disable the action (see 3.1.3); and 5) effects, which store the results of performing the action.

Besides, when no action is selected, it results in the player character “reading” its description. This description can be edited in the “Documentation” tab. If the game is in first person, the description will appear near the center of the screen.

The actions available for items (Figure 37) are described below.

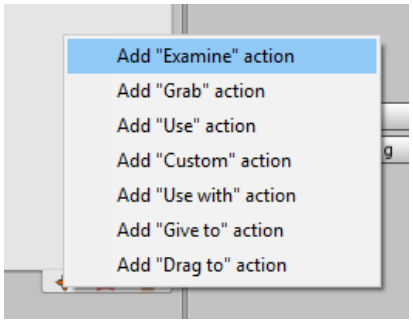


Figure 37. Possible actions.

2.4.2.1 Examine

When an object is examined, by default, the detailed description of the object is shown in the screen. Even when not added to the item, the “Examine” action will be available to the player. However, the “Examine” action can be reconfigured and this is achieved by adding the action to the list and creating a new behavior for it.

2.4.2.2 Grab

Once the “Grab” action is defined for an item the player will be able to grab the object from the scene and place it in the inventory. This item can later be used or combined with another object, as well as given to another character. This default behavior is added to the effects when the action is created, consisting in a “Remove element” effect (that removes the element from the scene) and a “Generate item” that adds the item to the inventory. These effects can be changed by clicking in the “Effects” button of the action.

Note: Clarification about how “Grab” action works with the element references.

The same object can be included in different scenes. When the user performs a “Grab” action the game engine removes the object from the scene and adds it to the inventory.

This action affects the whole chapter, that is, the object will not be shown in the chapter even it is referenced in other scenes (more information about in section 3.1.10).

Moreover, when a "Use" and "Grab" actions are defined over the same interactive element the engine will only display the action "Grab" when hiding the action "Use" is not active, it means, when the conditions over "Use" actions are not met.

2.4.2.3 Use

This action defines that the item has a utility that does not require another character or object. This use can be different for every object and will have unique consequences that must be explicitly defined.

2.4.2.4 Use with

This action, although like "Use", requires the intervention of another item. When our item is used with the "target" item, the effects of this action will be evident. However, using the item with an item different from the target will have no consequences.

2.4.2.5 Give to

"Give to" behaves just like the "Use with" action, but the target of this action must be a character. Besides, this action is only available for items once they have been grabbed and placed in the inventory. By default, after an element is given to a character, it will disappear from the inventory and the other consequences made evident.

2.4.2.6 Drag to

This action, defined just like "Use with", only differs from the former in the way it is used by the player during the game. The *effects* of this action will be triggered when the item is dragged onto the target item or character.

An infinite number of actions can be defined for the same object. When more than one action of a certain type is defined (for instance, two "Examine" actions), the engine will choose which to use based on the *conditions* (these can be edited in the actions list). The rule followed is that the first action whose conditions are satisfied is used.

The column named "Needs to reach", which is only relevant in third person games, allows for actions to be restricted to when the player's avatar can reach the item in the scene. This allows for actions such as "Shout to" that can be performed from anywhere and others (such as "Grab") that need the player close. If this option is checked, barriers (see 3.4) or other obstacles can make it impossible to perform the action.

2.4.2.7 Custom

The custom action can define any new action that requires a custom text, buttons or animations. If the custom action is added to a character it will be only a simple action, but if it is added to an item it can also be an interaction such as "Use With" or "Give To" meaning that it will require a target. When selected, the action list will show a resources menu on the bottom with a appearances editor (for multiple appearances e.g.: when using multi-language). Each appearance includes the fields:

- Button:
  - Image for normal button: The image displayed when the button is idle.
  - Image for hovering button: The image displayed when the button has the mouse or pointer on top of it.
  - Sound associated: The sound played when the button is hovered.
- Animations:
  - Looking right use: Animation played by the player for when the action is performed looking right.
  - Looking left use: Animation played by the player for when the action is

performed looking left.

2.4.3. Information

Just like for scenes, the information about items can be modified in the “Information” tab (Figure 38).

There are 4 fields and one checkbox that can be modified by the user:

- Documentation: this field is used to document (i.e. it is not used during the game) the item. Information in this field is stored in the game but only used by the game editor, to provide further information about the item when edited by someone else or at a different time.
- Name: This is the name of the item. This name will be shown to the game user when the mouse cursor is over the item.
- Brief description: This description should be concise and provide an idea of the nature or possible use of the item. The player will be able to see this description by left-clicking the item in a scene.
- Detailed description: This description can be longer, providing additional details about the object. By default, this description is shown to the user when “Examining” the item. However, such default behavior could be overridden.
- Object returns to its origin: This checkbox establishes how the item will behave in the case of being dragged. If it is selected, items will return to their original position once the player releases the mouse button.

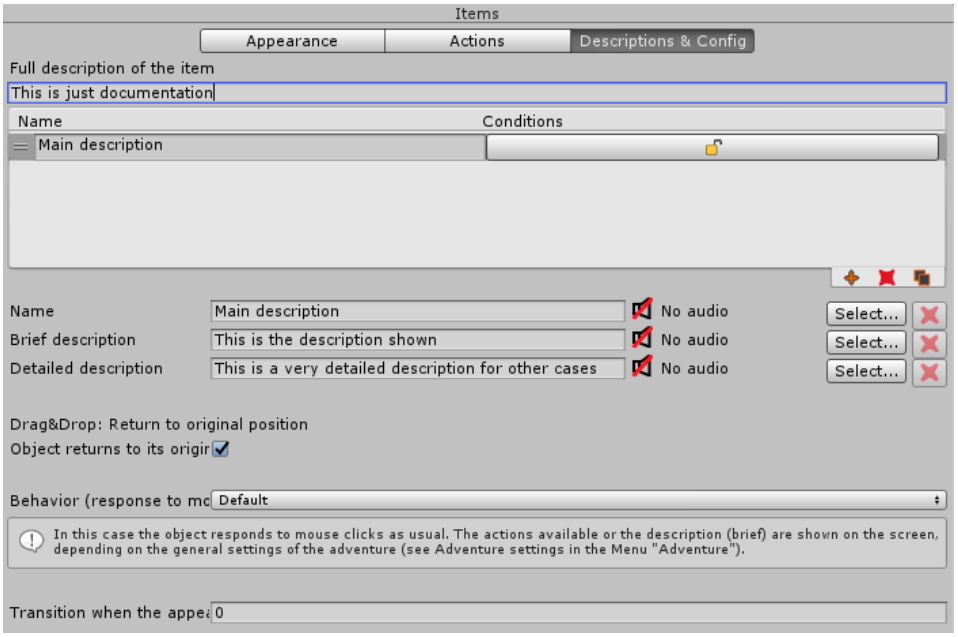


Figure 38. Items "Descriptions & Config" tab.

2.4.4. Inventory

In uAdventure the Inventory appears in game as a backpack icon by default. When the player clicks the icon, the inventory pops up on the whole screen and can be closed in the top-right corner (Figure 39). In this inventory, all the items will be displayed in the order they have been gathered (using the grab action) and the user can scroll from top to down by using the scroll wheel in the mouse or clicking in any part of the inventory (even on top of the items) and dragging the inventory

up or down.

It is possible to replace the icon and change the inventory behavior style, appearance, position, and more in the “Inventory” section of the adventure configuration window (see section 7.2.4). However, regardless the configuration, the inventory can be removed from scenes individually by using the “hide inventory” parameter in the appearances window.

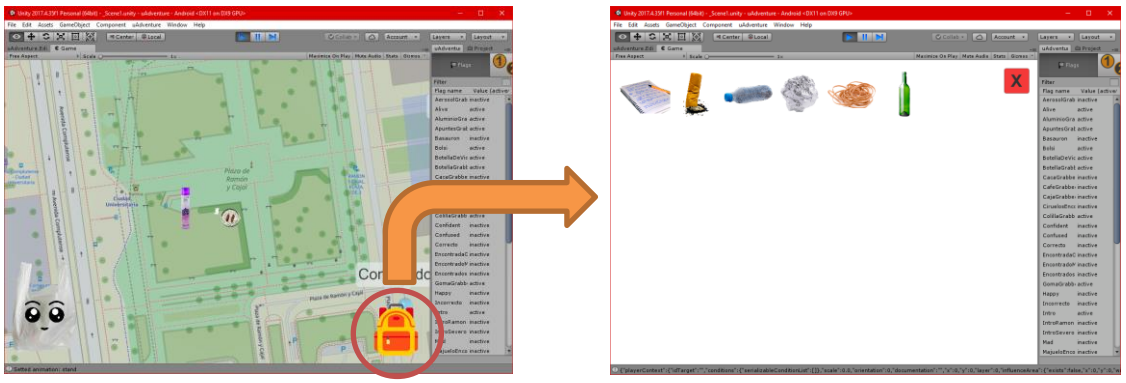


Figure 39. Inventory behavior.

In addition to adding items to the inventory with the “grab” action, the items can be removed from the inventory by using the “use with” and “give to” actions. Additionally, you can use effects to add or remove items. By using the “generate effect” you can add items to the inventory, and with the “consume effect” you can to remove them for it. In fact, when a “grab”, “use with” and “give to” action is added in the item using the uAdventure editor, uAdventure preconfigures its effects to perform such effects.

2.4.5. EXAMPLE: Creating an item with different views and actions

According to the story, you are in your office and get a phone call, so we now must make that phone in your office ring and you will have the opportunity to answer that call.

First, we are going to create the phone and add, by default, the resources of the phone ringing. We can add another resource set by clicking on the add button (✚) on the top white list and once you select the “Resources 2” you will see that the image below changes automatically (Figure 40). Select the idle phone for the second resource pack and select the “Resources 1” back as default.

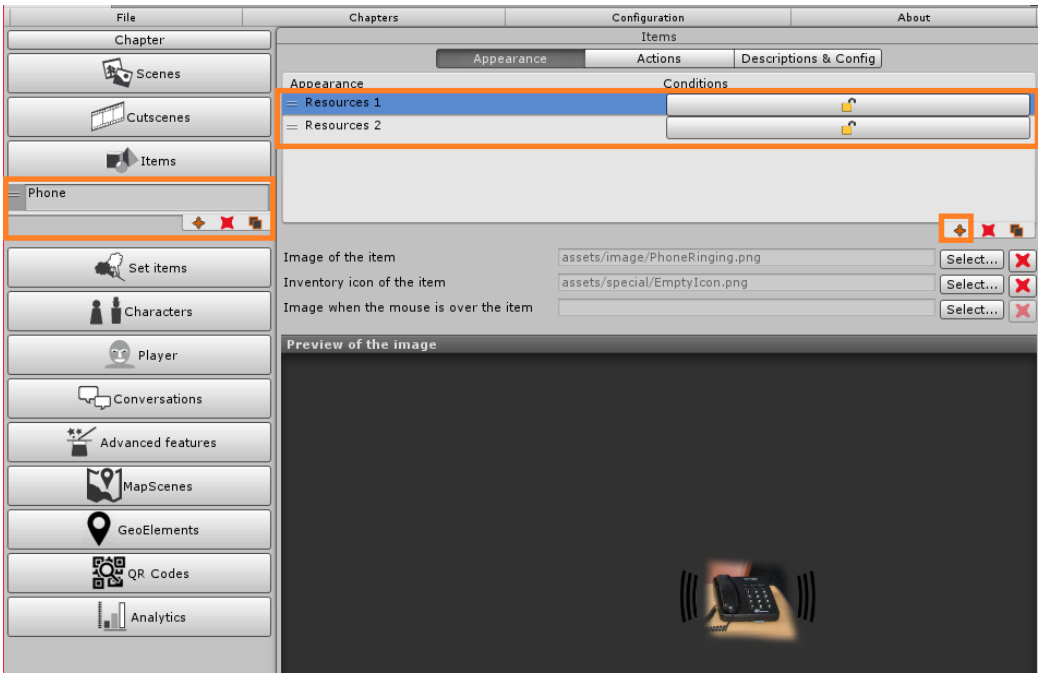


Figure 40. Phone item with two different resources for ringing and idle states.

Now that we have our appearance configured, we can add some interactions to it. As we mentioned, the user is going to be able to respond our phone call where he is going to get notified about the fire. To add this action, switch to the actions tab, click in the add button (✚) below and select “Add use action”.

Now we have our actions ready but, without adding any effects, the action will be meaningless. If we think about the action purpose, the player is responding to a phone call. Therefore, we should represent the phone conversation as it happens. One way could be to create a conversation and show it up when using the phone. We will explain conversations later in this guide. Another option is to create another slide-scene instead using two frames for it. Create the animation “PhoneConversation” as in the previous examples and connect it with the “PlayerOffice” in the end (as in Figure 34), to get back to the office as the phone ends. The frames used will be the ones in Figure 41. Note that animation names must be unique in the project and also cannot be the same as any identifier in the game.

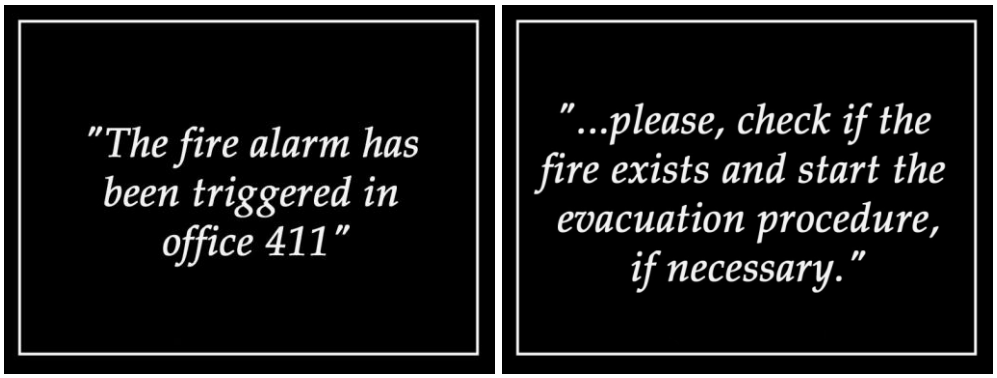
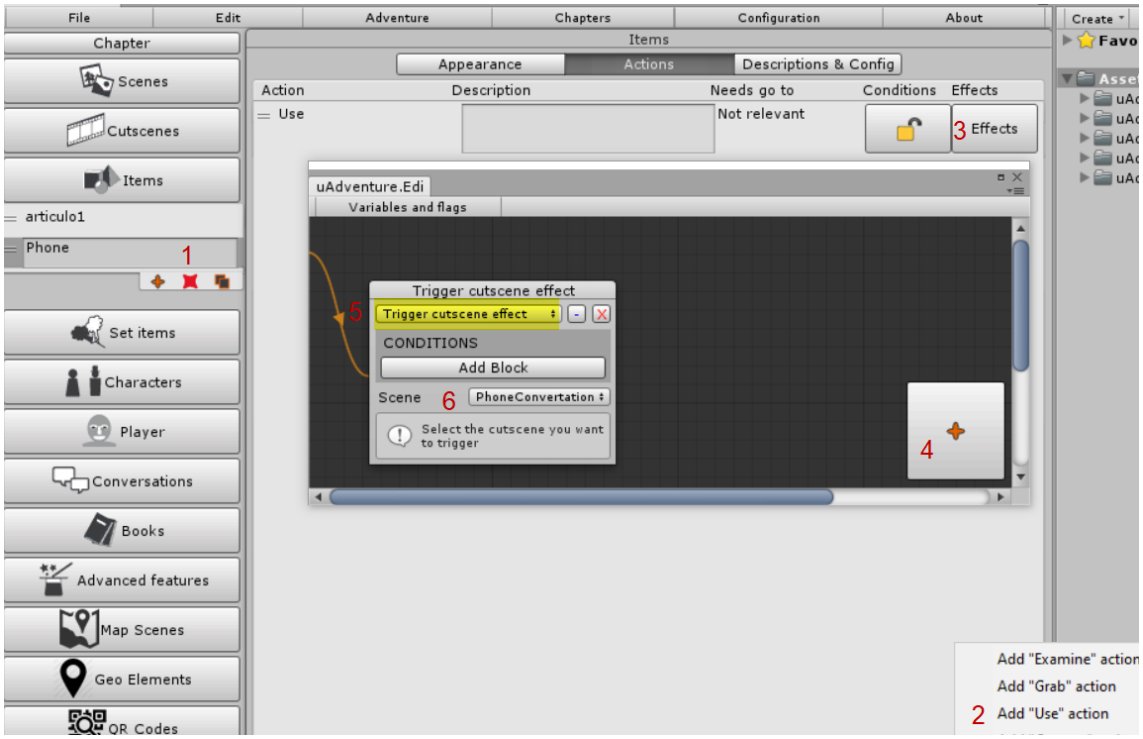


Figure 41. Phone call frames.

Aquí encontre un detalle que es importante explicar pues parece obvio pero no lo es y esque se ocupa usar el **Trigger cutscene effect**



Later in this manual, after both conditions and effects are explained, we are going to connect all the pieces together and make the phone switch from ringing to idle and show this animation.

2.4.6. EXAMPLE: Adding an item to a scene

Once an item has been created, it can be referenced from anywhere in the chapter. For instance, a reference to the item can be added to one or more scenes. This example shows how to add an item to a scene.

The first step is to select the relevant element from the left panel or structure panel which is going to be our “PlayerOffice” scene. To place the item, we must add a new reference to it, so we must go to the “Element references” tab of that scene in the right-side panel. Now the center panel shows a list of references on top and a preview of the scene at the bottom. Clicking the add button () next to the reference list allows to add a new reference of one of the three element types in <u-Adventure>: items, NPC (Non-Player Characters) and set-items (Figure 42).

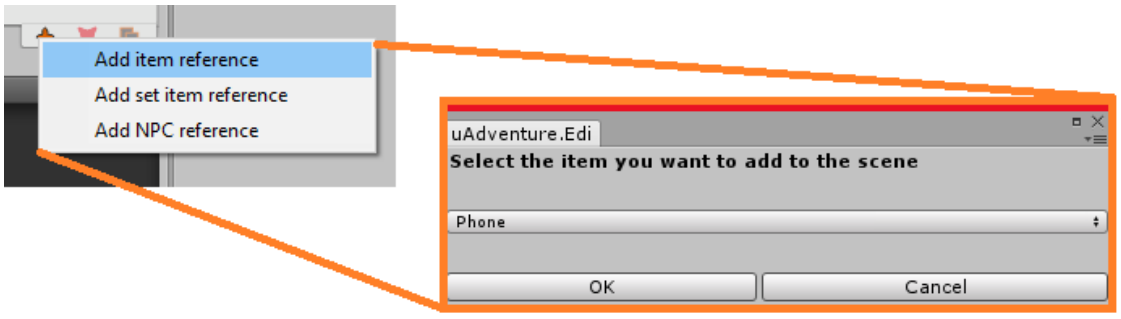


Figure 42. Add item prompt.

To add an item to the scene, choose that option. This will prompt a dialog asking for the specific item that we wish to reference by providing a list of elements of the selected type (Figure 42). In our example, we will choose the recently created “Phone” item and then click the OK button. The item will be added in a default position (Figure 43).

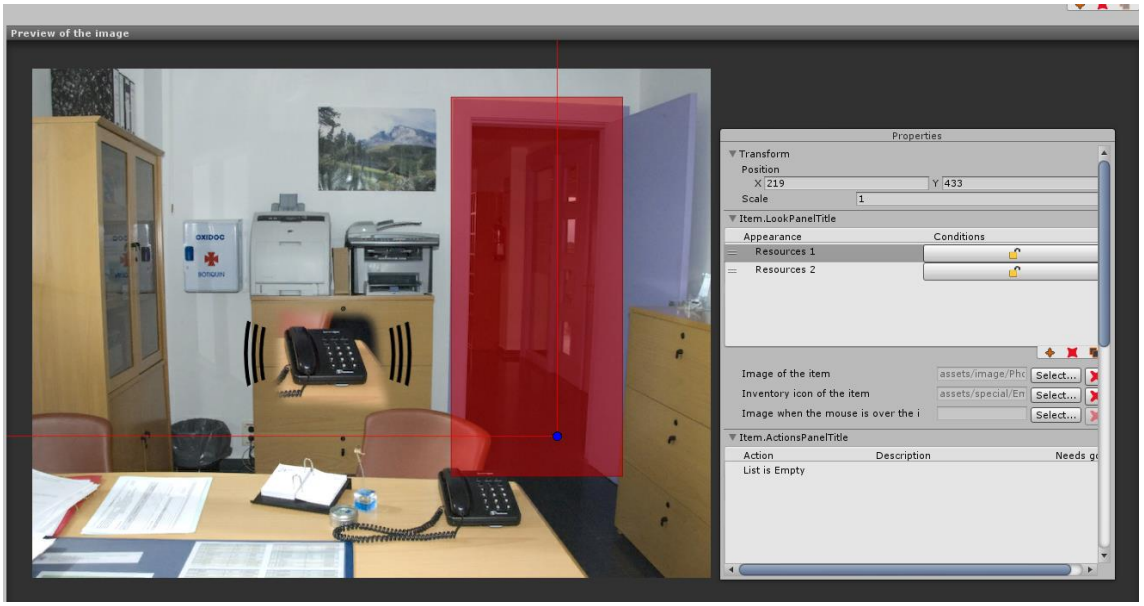


Figure 43. Scene edition panel, "Element references" tab just after adding a new element reference.

As most elements in <u-Adventure>, element references can be scaled and moved using drag-and- drop. Changes in the element reference only affect that reference and not the item itself. Besides, this panel includes fields (just on top of the preview) to modify the position and scale of the references with higher precision. This allows the phone to be correctly placed over the table in our example (Figure 44).

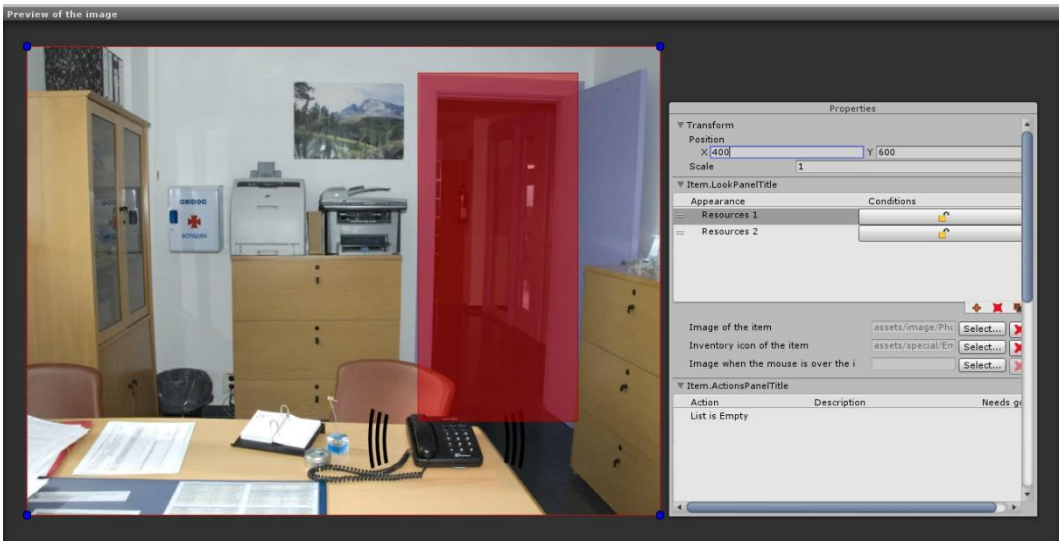


Figure 44. Scene edition panel at the "Element references" tab. After moving and scaling the "Phone", it now sits correctly over the table in the scene.

Other settings are available in the reference list:

- Layer: The layer is used to provide a sense of depth. A lower value means that the element is further from the viewer (more information about this can be found in section 3.2.1.
- Referenced element: This column shows the actual <u-Adventure> element that is being

referenced.

- Condition: This is the condition that establishes when, during the game, the reference will be visible to the player. If the conditions are true, the reference is shown and if they are false the element is hidden. Conditions are edited as any other in <u-Adventure> (see section 3.1).

In third-person games with scenes with trajectories, this panel also allows the edition of the "influence area" of an element reference (i.e. the area where the player must be as to be able to interact with the object). This, however, only affects scenes with trajectories and is studied in detail in the "Player movement", in section 3.5 of this guide.

2.5. Set-items

Set-items only provide visual information and cannot be interacted with.

2.5.1. Creating a new set-item

Just like regular items, set-items are added by clicking the add button (+) after selecting "Set items" in the structure panel (Figure 45). The new set-item will have a default ID that can be changed by clicking the "rename" button, with the same restrictions as other ids in <u-Adventure> (starts with letter, unique, etc.).

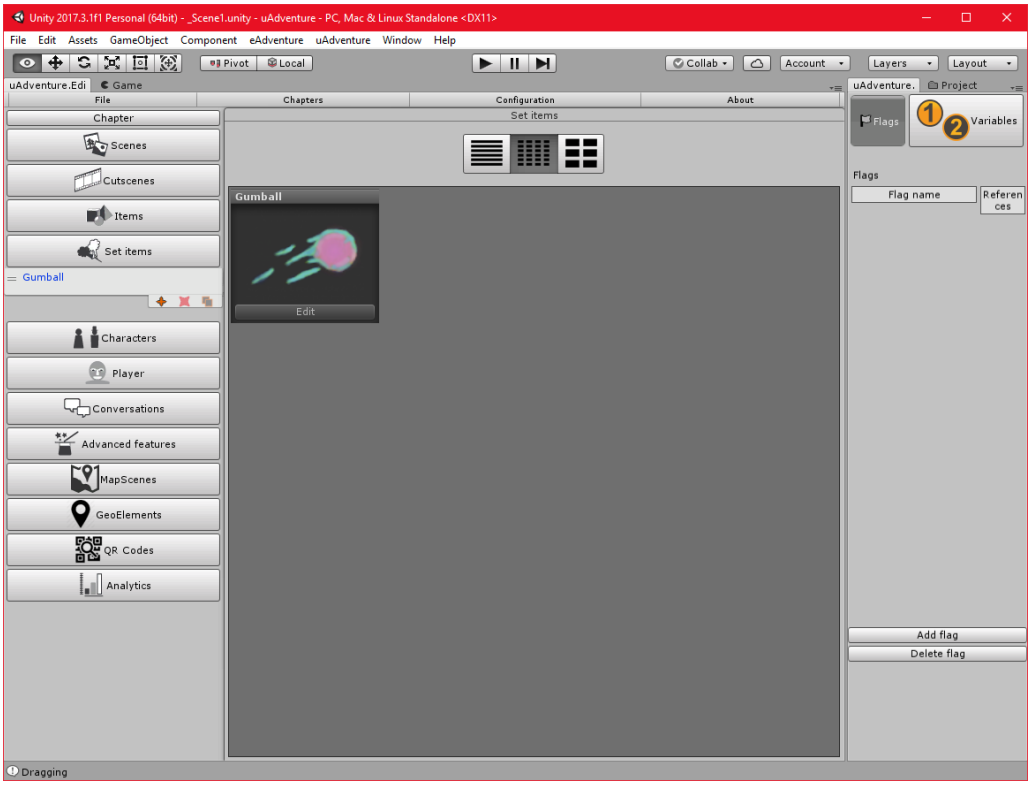


Figure 45. "Set items" general view, with just one set-item in the chapter.

The edition panel for set-items is like the one for items but with fewer tabs. The "Appearance" tab is used to choose an image for the set-item, but in this case no icon is needed (Figure 46). There are no restrictions to the size the image of the set-item must have.

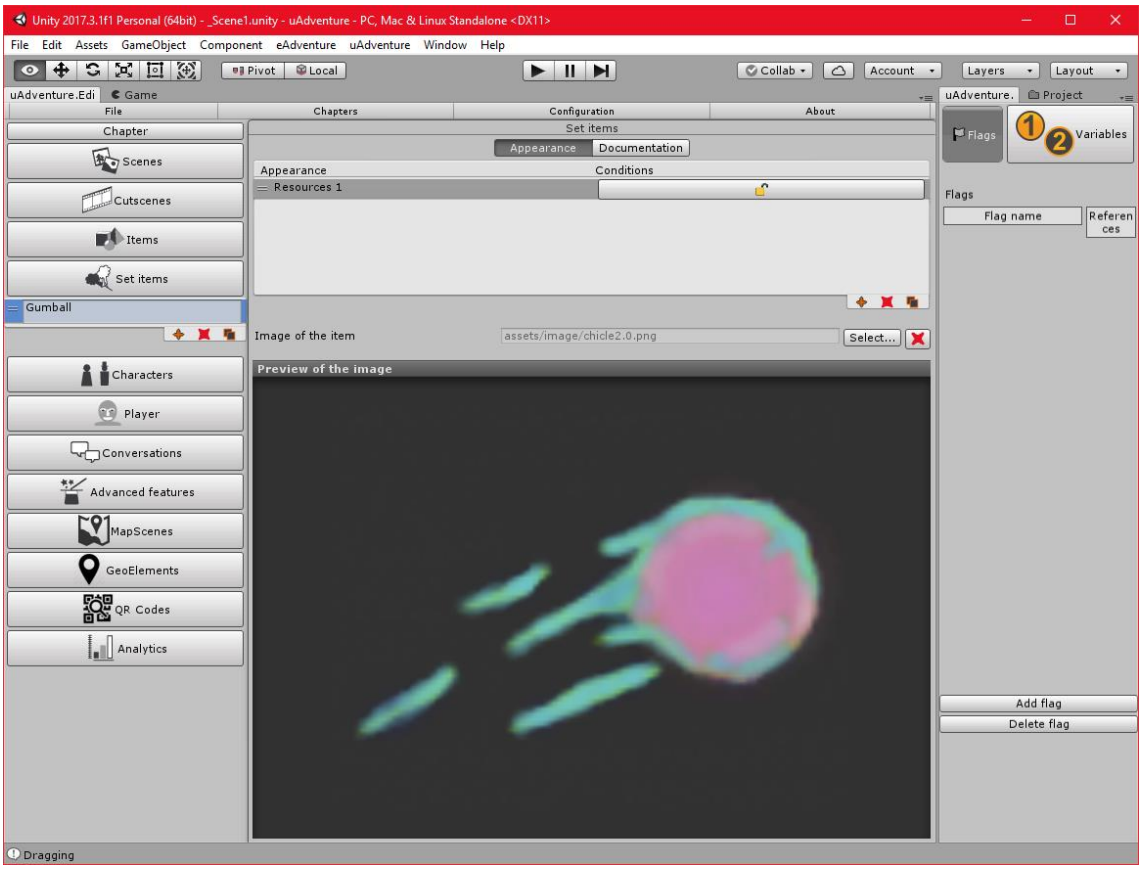


Figure 46. Set-item edition panel in the "Appearance" tab after setting the image.

2.5.2. Information

Just like for regular items, set-items can be documented in the "Information" tab. There is also a "Name" field, this is not used during the game and is also used for documentation purposes.

2.5.3. Adding a set-item to a scene

This process is the same as the one for regular items (see 2.4.6), just that instead of adding an "Item reference" we add a "Set-item reference". Set-items can be moved and scaled in the same way too.

2.6. Books

Books are included in to provide large amounts of information in a natural and easy to access manner. Books can be displayed at any time during a game and are of two basic types:

- Simple books: This books are edited directly in the editor and are made up of paragraphs, titles, unformatted text, bulleted lists and images. The contents are spread into different pages as needed and are easy to create

2.6.1. EXAMPLE: Create and edit a simple content book

The first step is to create a new book of this kind. To do this, we select “Books” in the structure panel and then click the add button (). When asked for the kind of the new book, we choose “Simple content book”. The new book edition panel will appear (Figure 47).

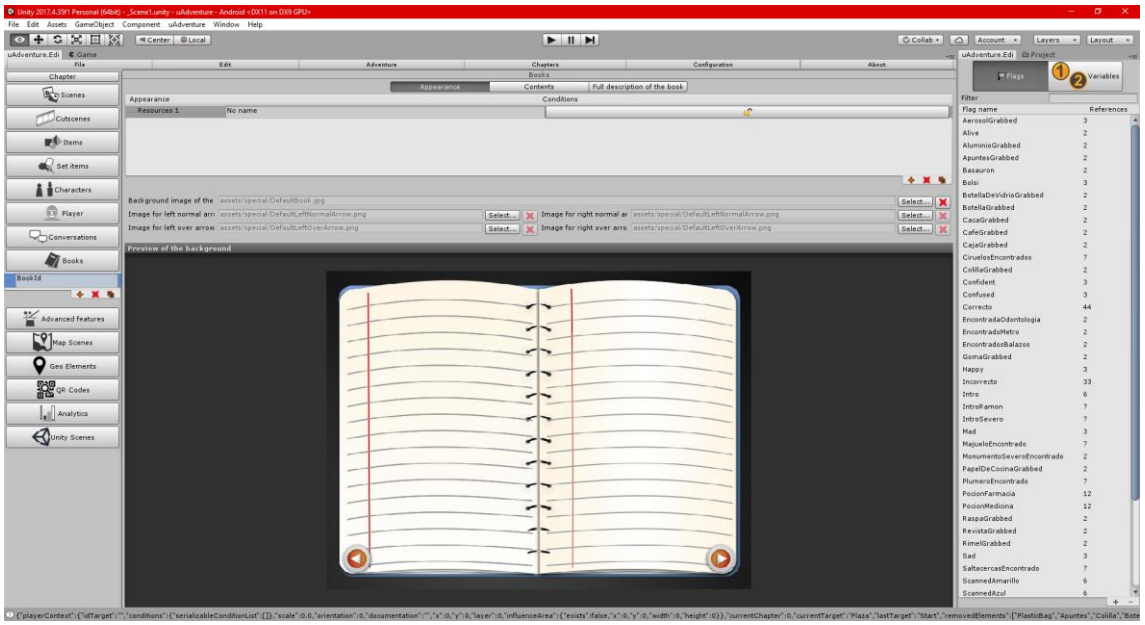


Figure 47. Simple content book edition panel in the “Book content” tab

There are three different tabs in this book edition panel:

- Book content: the contents of the book (paragraphs, images, etc.) are edited in this tab.
- Appearance: the looks (i.e. graphic assets) of the book, including the background, arrows, etc. are edited in this tab (Figure 47).
- Documentation: this tab allows for the documentation of the book for future reference and the information here is not used by the game engine.

In this example, we will create a book that has a title, a paragraph and an image. This book will provide information about raspberries. To do this we must first select the “Book content” tab.

The edition panel has to main sections. In the bottom we have a preview of the book that we are creating. At the top we find an ordered list of all the elements (paragraphs, images, etc) that the book has. The buttons to the right of the list can be used to add, remove and sort these elements. To add a title we click the add button () and choose “Title” from the options that appear. We then modify its contents directly within the list, change it to “Raspberries” (Figure 48).

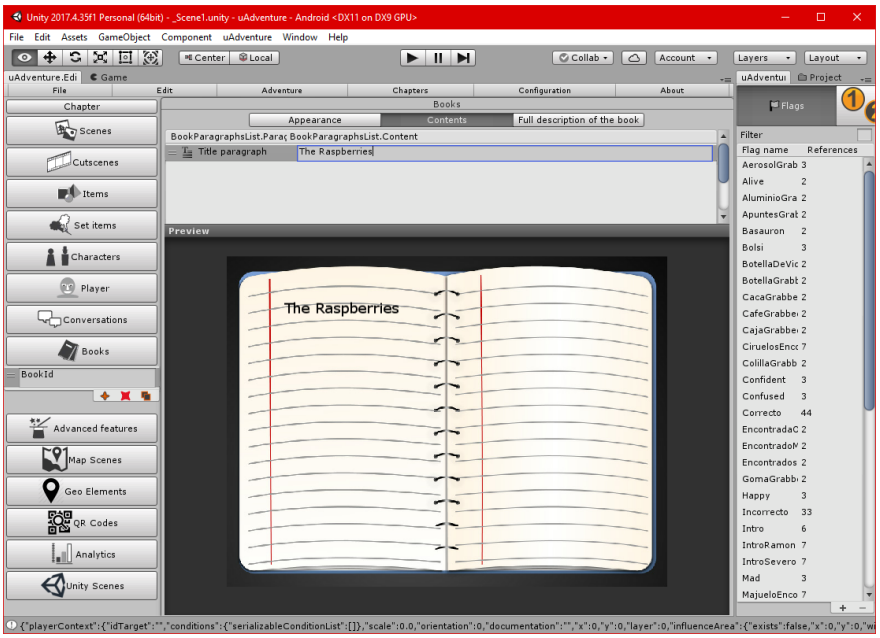


Figure 48. Simple content book edition panel, after adding a title element.

After this, we repeat the process for a text paragraph. The contents of this paragraph can, for instance, be copied from Wikipedia. Finally, we add a new image paragraph and choose an image of some raspberries (Figure 49).

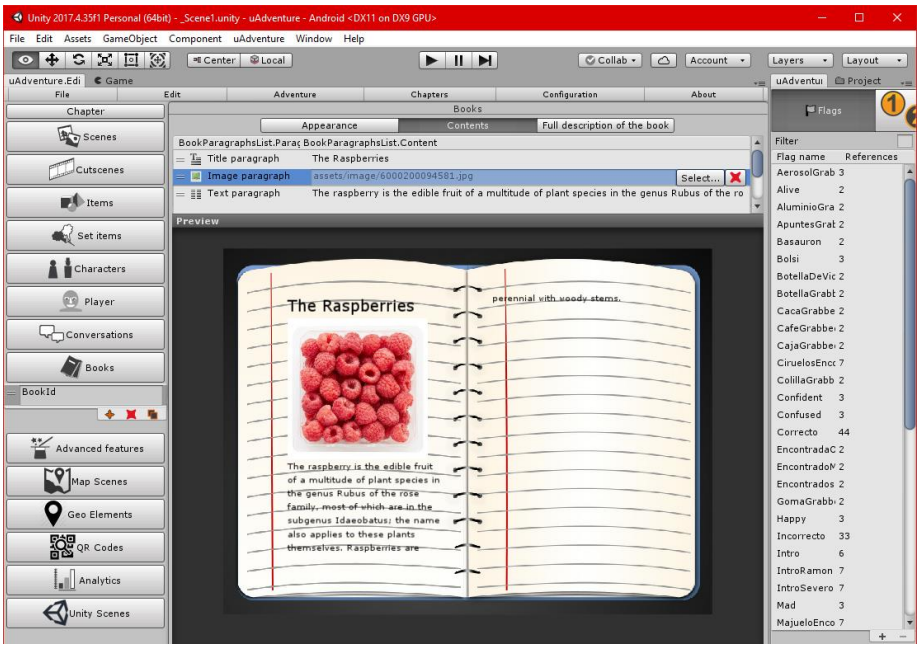


Figure 49. Simple content book edition panel, after adding an image paragraph.

The preview allows us to see the result as it is to be shown in the game. We can drag the elements from the left (=) symbol to reorder them. For instance, we could place the image before the text paragraph.

Finally, we could use a “trigger book effect” (section 3.1.10) in game to explore and see the contents (Figure 50).

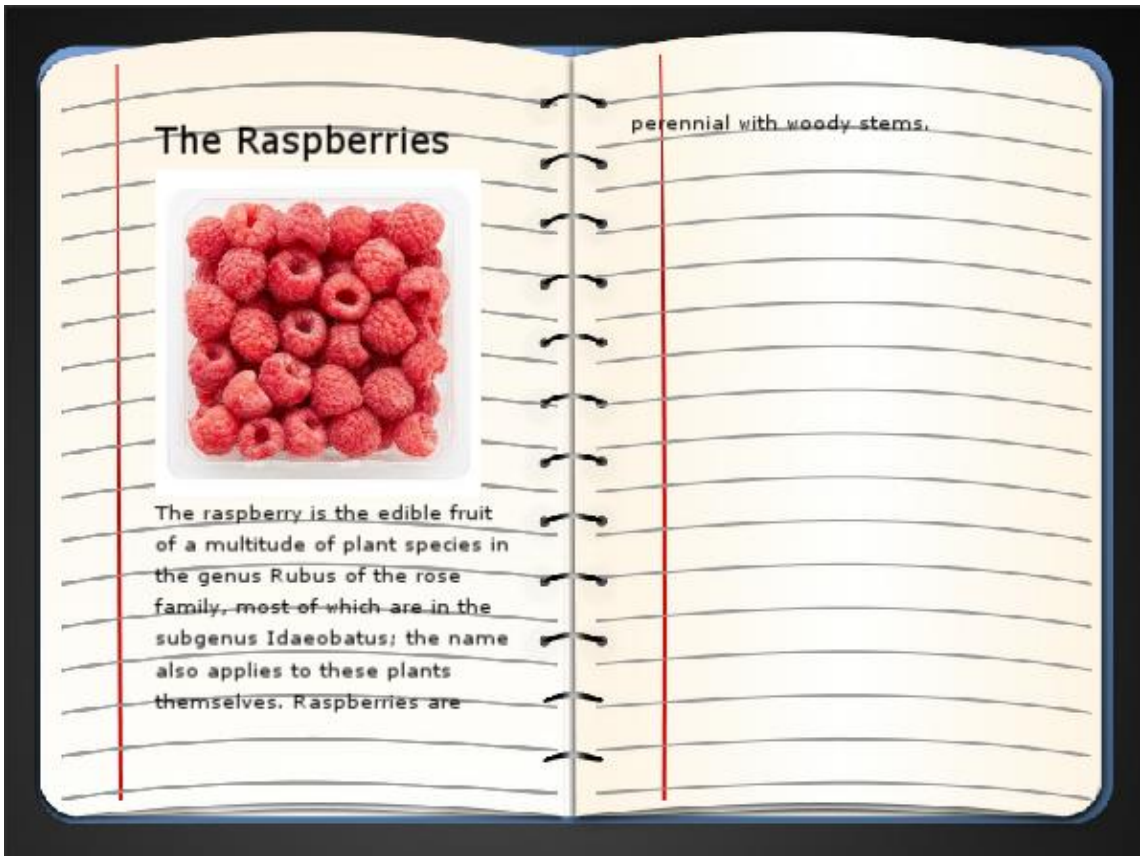


Figure 50. Simple book in-game view.

2.7. Characters

Characters are game elements which, among other things, the player can talk with. Characters (also referred to as NPC or Non-Player Characters) have several differences to the items from the edition point of view. In the first place, characters can be assigned animations, providing them with life-like characteristics. Moreover, some actions are different given that characters cannot be grabbed, used or given to, but they can "receive" an item and can be talked to. However, characters can be dragged.

2.7.1. Create a new character

New characters are added following a similar procedure to other objects. To do this, see sections 2.2.1 and 2.4.1.

Once a character is added, some difference become apparent. The appearance of this sort of element requires more resources. This means that different animations must be provided for different situations (talking, grabbing, and walking) and with different orientations (Figure 51).

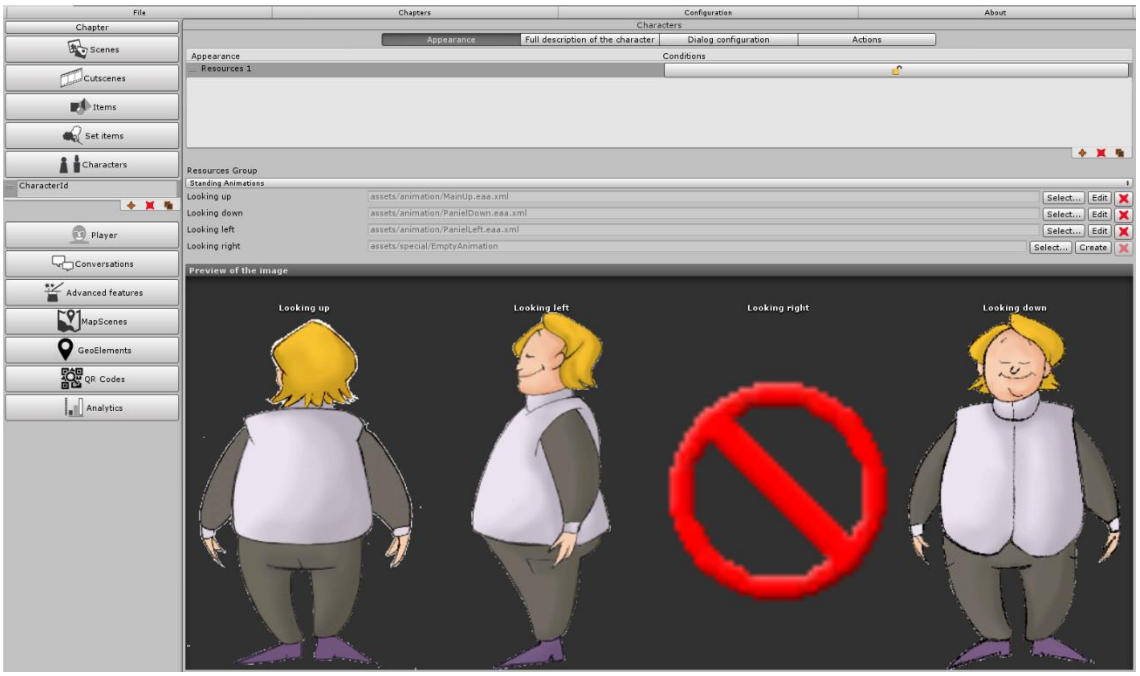


Figure 51. Character edition panel in the "Appearance" tab allows for the edition of the characters animations.

Several features of animations in <u-Adventure> must be detailed. Animations are a set of images, drawn one after the other. This sequence is defined by the frames and transitions of the animations. You can learn more about animations in section 3.8.1.

The animations that must be configured for a character are:

- Standing animations: looking up, looking down, looking right and looking left (this last one is optional).
- Talking animations: speaking up, speaking down, speaking right and speaking left (this last one is optional).
- Using animations: object to the right, object to the left (optional).
- Walking animations: walking up, walking down, walking right and walking left (optional).

When an action animation does not have a visual representation for its left view, a vertically mirrored version of the right animation is used instead. The same happens when the right view is missing but the left view is present.

2.7.2. Character dialog configuration

This panel has several options that affect how the player speaks. The first set of options refers to how the text will be displayed in the screen (Figure 52):

- Show speech bubble: If selected, the dialogues of the character will be shown inside a bubble just as in comic books.
- Font front color: this button allows for the configuration of the front (or center) color of the font. Clicking the button will open a color chooser.
- Font border color: this changes the color for the border of the font.
- Bubble background color: this button changes the color for the background of the bubble. During the game, this will be filled with a semitransparent tone.
- Bubble border color: this button changes the border of the bubble.

Note: Althought TTS was one of the features of <e-Adventure> this feature is not yet available for uAdventure.

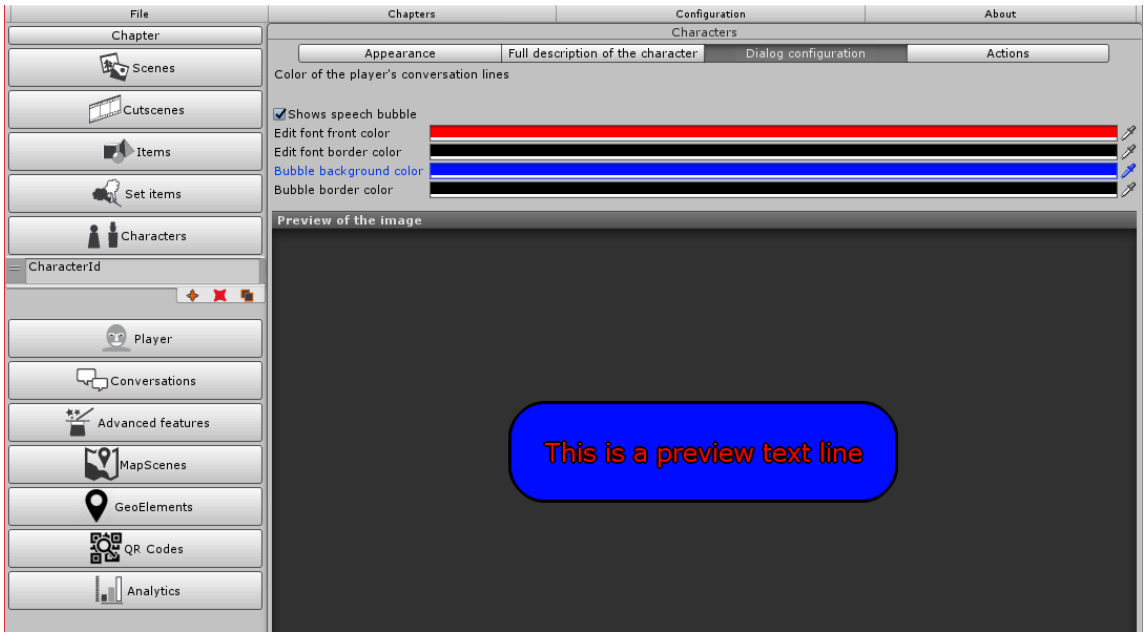


Figure 52. Preview of the text line of a character (or the player).

2.7.3. Adding characters to the scene

Characters are added to scenes in the “Element reference” panel. This is done just like for the other elements (see 2.4.6).

2.7.4. EXAMPLE: Creating a character to find and add it to another office

In this example, we are going to create a possible fire victim. One of the objectives of the game is to look for possible victims that have not heard the fire alarm or are not aware of the risk at all. First, we are going to create the character that is going to be Balta, the professor.

Go to the Characters section and press the add button () and rename it to “Balta”. Then, select it and go to the “Appearance” tab. Since this character is just going to stand idle in front of you, no moving or using animations are needed. We also make the talking animation to be the same as the idle one.

First, we are going to set the “looking down” animation by clicking on “Create” and rename it as “BaltaAnimation”. Once the animation editor is prompted, we will select the first frame and change it to the first Balta image. The result can be seen in Figure 53.

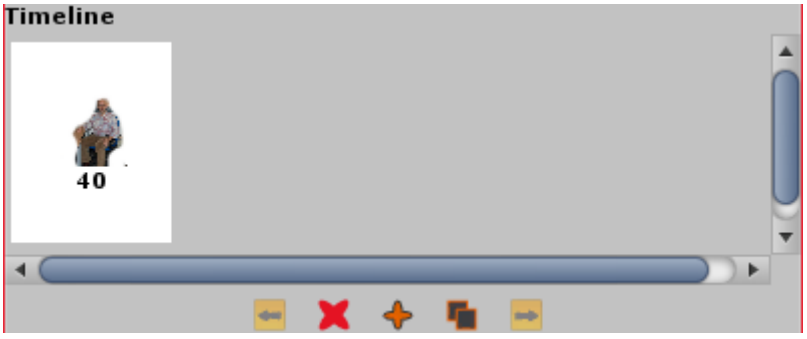


Figure 53. Balta standing animation.

Then, we will switch to the talking animations and, in the “Speaking Down” field we are going to select the previous created animation “BaltaAnimation.eaa.xml” on assets/animation folder. The final appearance of the character can be seen in Figure 54.

Figure 54. Balta character with looking down animation.

Now we are going to create another office and a few corridors towards it. The process of creating new scenes and connecting them through exits is the same as explained in 2.2.2 and 2.2.4. The resources we are going to use are the ones in Figure 55.

Figure 55. Resources for the different scenes.

Finally, in the next Figure 56 we can see the resulting scene exit flow.

Figure 56. Final chapter scenes to navigate to Balta office.

To finish our scenario, we can add Balta in his office. This is made exactly as in 2.4.6, adding a NPC reference this time to Balta character.

2.8. Conversations

Players may interact with characters as well as with items (see Section 2.4.2 on items interactions). The interactions with characters are significantly different than those with items. In the "Actions" tab of the characters menu, we find a specific action for characters: "Talk to". This sort of action requires the existence of conversations in the chapter. Conversations are dialogs between the player and one or more characters, and can be created for several purposes, such as: guiding the player, providing information, or evaluating the player.

Besides the player, multiple characters can take part in a conversation. Just like in classic *Lucas Arts*™ games, sometimes the player might be presented with a list of options to choose from. These options correspond with the following line the player will utter. The path followed by the conversation will change consequently.

Conversations should be bound to a character to be triggered when the player talks to it. To be able to dos this, the conversation element must be previously created by selecting "Conversations" in the structure panel and clicking on the add button (). The conversation will then be get bounded to the character by adding a "Talk to" action in the character's "Action" tab.

Conversations have a graph structure, made up of a set of nodes and the links between them. All conversations start at an initial node and follow a linear path (one node is "read" after the other). Bifurcations can be added to the conversation, adding special nodes named "option nodes". Circular paths can also be created to repeat parts of a conversation (e.g. until the correct answer to a question is given).

Nodes in a conversation use different graphic representations, depending on their type, and can be linked to each other. For each node, the left side represents the inputs and the right side, the outputs. If no inputs are present, this is the initial node and if no output, it is an end node. There are two node types:

- Dialog nodes. This node has dialog lines that will be read by the characters or player specified in the "Speaker" drop-down list, in the given order. The dialog will show all the lines inside of the node (Figure 57).

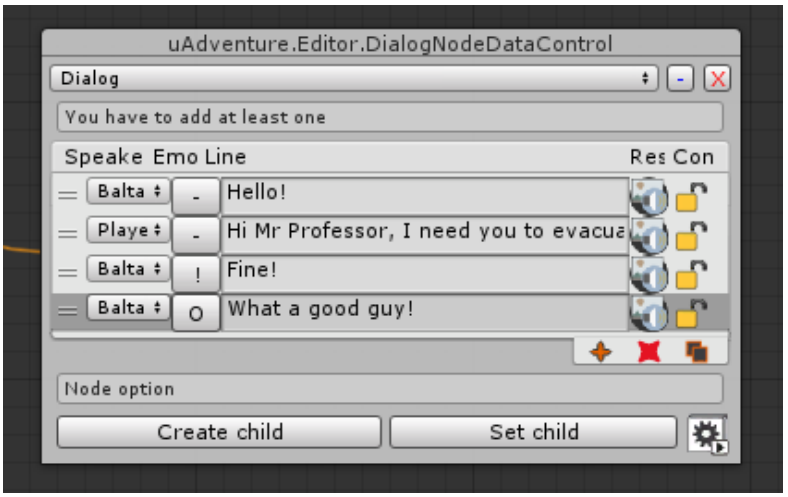


Figure 57. Dialog node.

- Option nodes. This node specifies the options that the player must choose from when that point in the conversation is reached. The selected option will be the next line of dialog that the player reads, and will determine the path followed by the conversation. They look very similar to the dialogue nodes, except they show "OPTIONS" as title **(Figure 58).**

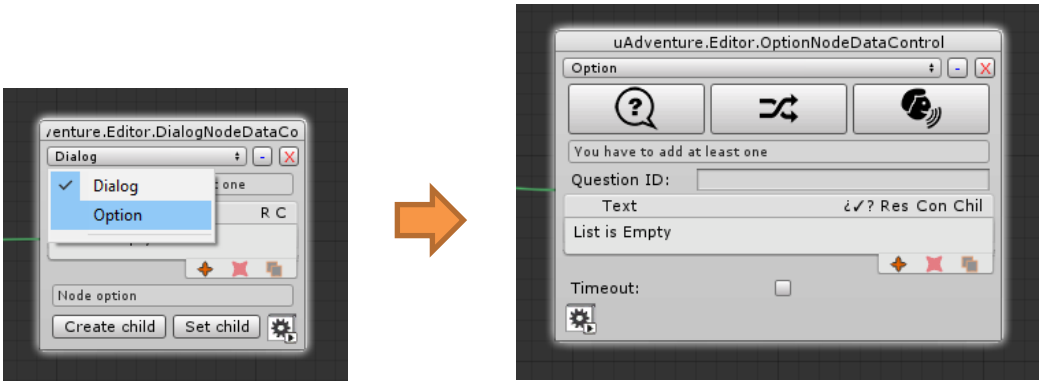


Figure 58. Options node with three children.

All the nodes have the different options:

- Top title: Can be either DIALOGUE or OPTIONS.
- Type selected: Allows changing the type of the node between the dialogue and the options (Figure 58). When changing the type from dialog to options the first child is preserved and the node is created with one option. On the other hand, if an option node has multiple children when changing to dialogue, an alert will be shown to inform the user that only the first chill will be preserved.
- Collapse button: Any node in a conversation can be collapsed by clicking on the "-" symbol on the top right of it (Figure 59). When a node is opened you can see the different contents it holds depending on its type.

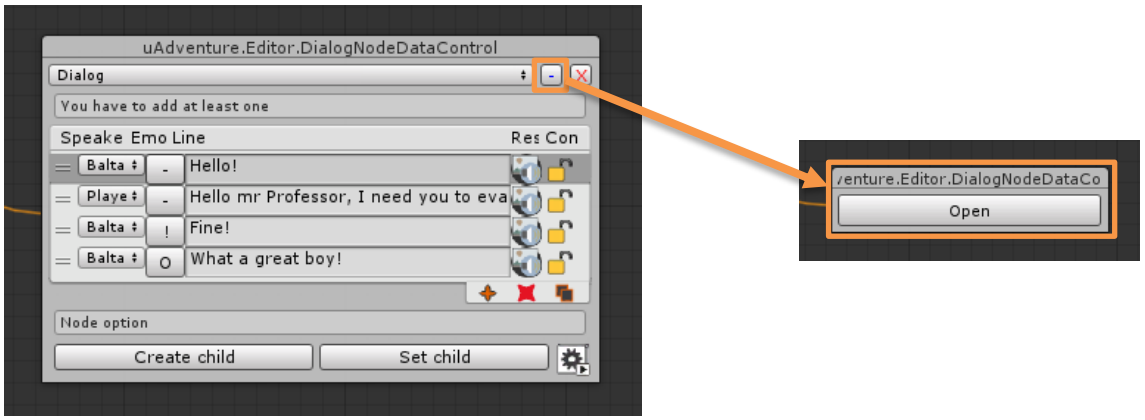


Figure 59. Collapsing node process.

- Delete button: By clicking on the "X" symbol, it will delete this node but preserves the first children. When deleting an options node an alert is displayed informing that the rest of the node children will be deleted if the user continues.
- Node effects: On the bottom right, a gear icon ( ) allows to open the effects associated with that node. This is used to define the block of effects that is triggered after all the lines in the node are read. This are game effects, just like the consequences of actions (see 3.1 for a detail explanation). When effects are enabled, the gear symbol turns green ( ).
- Node resizing: By hovering the mouse on the right side of an open node the pointer will show the width resize symbol ( ). Clicking on the right side of the node while the resize

icon is shown and dragging the mouse allows you to change the node width.

The dialog window also allows to easily organize and manage our nodes. First, nodes can be selected and moved in the graph by clicking on them. When a node is selected, its border will be highlighted (Figure 60). Also, when a node is not collapsed, it can be expanded to the right by clicking on the right side of it when the "<->" symbol is shown. Finally, multiple nodes can be selected by clicking and dragging the mouse inside of the window as in any file explorer (Figure 61).

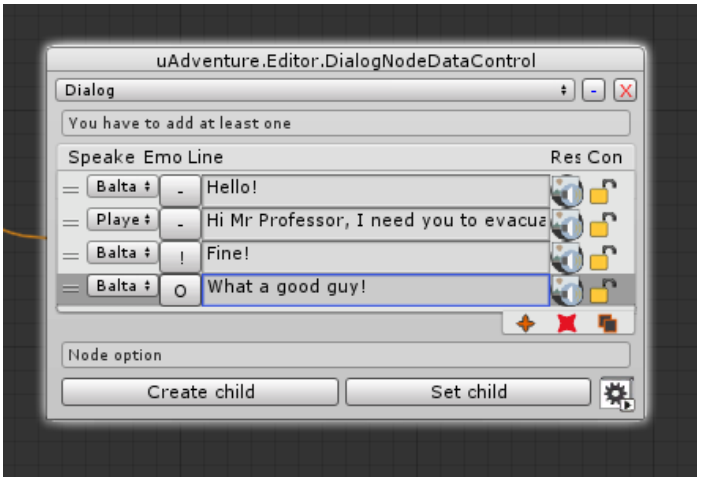


Figure 60. Selected node dialog box.

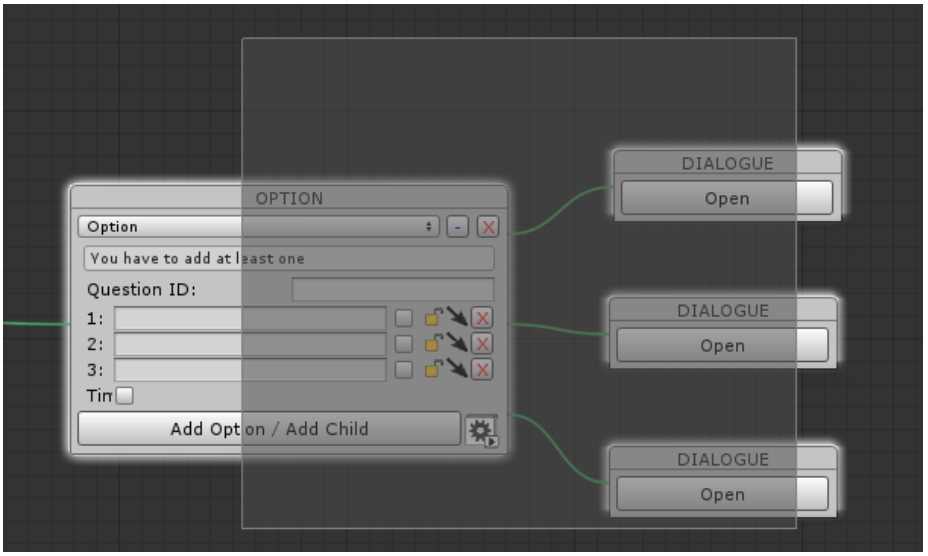


Figure 61. Multiple node selection tool. When multiple nodes are selected, the drag is applied to all of them at the same time.

2.8.1. Dialog node contents

Examining a dialog node in detail we can see different parts:

- "New line" button: When clicked adds a new line to the dialog node.
- Speaker column: Identifies the actor that is going to read the line, among all the characters in the chapter or the player.
- Emotion column: Identifies the different emotions (Figure 62) that the speaker can display. The emotions are represented by an icon being: (-) Normal; (!) Yell; and (O) Thought.

Figure 62. Different emotions. From top to bottom: (-) Normal; (!) Yell; and (O) Thought.

- Line: The text to be talked by the character.
- Resources symbol: The resources symbol ( ) identifies the different resources for the line, allowing the message to include an image inside, an audio or being synthetized by a machine generated voice. When the message includes resources the icon changes to a more colored version of the symbol ( ). When clicked it displays the resources window (Figure 63).

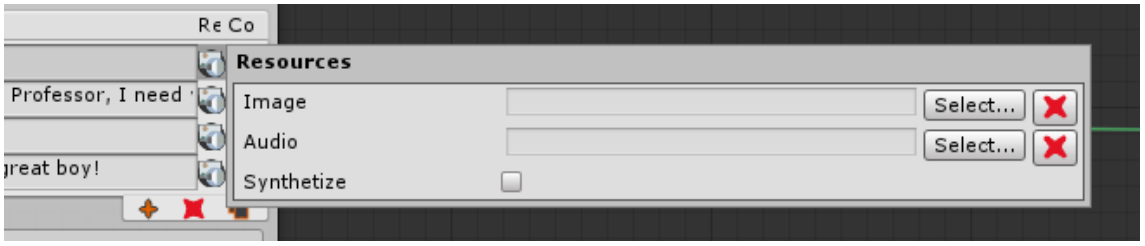


Figure 63. Resources for a line including image, audio or voice synthesizer.

- Lock symbol: The lock symbol ( ) identifies the different conditions for that line to show. When conditions are used ( ), the line will only appear if the conditions match (see 3.1.3).
- Delete symbol "X": When clicked, it deletes the current line.
- "Create child" button: This button will only be enabled if the current node does not have a linked child. When clicked it will create a new child for the node and link it.
- "Set child" button: This button allows to change the current child node to any other node (even previous nodes to make cyclic conversations). However, when a child node is completely isolated it gets removed. When clicked (Figure 64), the mouse will be followed by a blue arrow and the child will be changed when clicking another node. By clicking in the (+) button, a set of options to create nodes, remove the link and cancel the current assignation are displayed. In contrast to removing the child (that maintains the first child) by creating a new child when setting a child this removes all the other children.

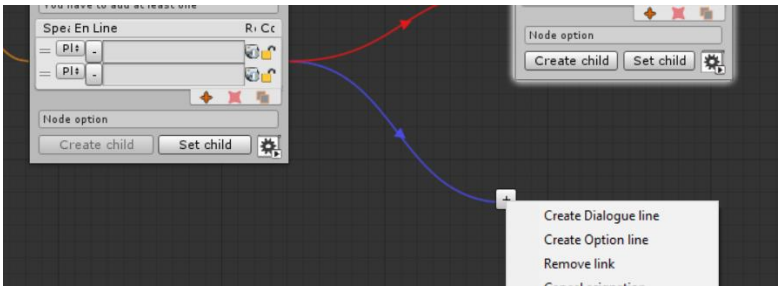


Figure 64. Setting up the child of a node in blank space.

2.8.2. Options node contents

Examining an options node in detail we can see different parts:

- Show previous question: Identified by the ( ) symbol when activated it displays the previous message as question in the dialog.
- Randomize options: Identified by the ( ) symbol, when activated it displays the options in random order.
- Show answer: Identified by the ( ) symbol, when activated it shows the selected option as the next conversation bubble.
- "Add option/Add child" button: When clicked, it creates a new option and a new child node.
- Option index: The number of the option. It can be used to identify the child node as all the node exits correspond, in order, with each option.
- Option: The text to be shown in the option that is going to be spoken by the player.
- Lock symbol: The lock symbol ( ) identifies the different conditions for that line to show. When conditions are used ( ), the line will only appear if the conditions match (see 3.1.3).
- Mouse symbol: Allows to change the option child when clicked. The behavior is the same as in the "Set Child" button of the Dialog Node (Figure 64**¡Error! No se encuentra el origen de la referencia.**).
- Delete symbol "X": When clicked, it deletes the current option.
- Timer checkbox: Options nodes can have a timer option. The timer will be shown on top of the question as a countdown. When enabled, next to it, it will show a number field to write down the timer length in seconds. The timer option also has a child node as any of the other options, which will be followed when the time runs out (Figure 65).

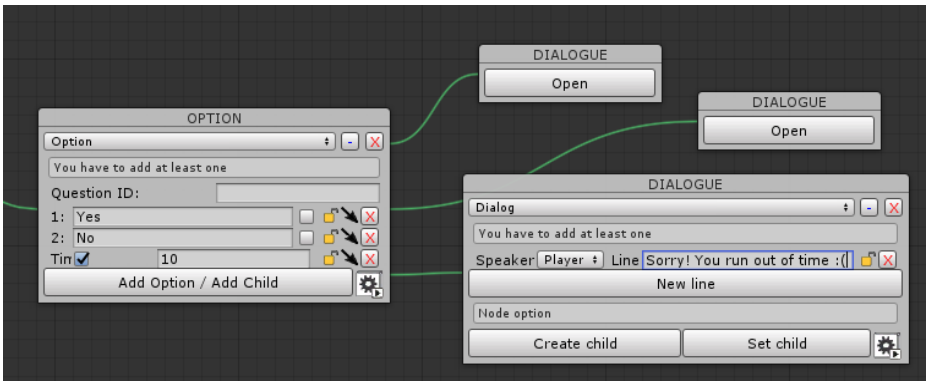


Figure 65. Timer option enabled and child dialogue node.

Notice that these new options added in <u-Adventure> simplify the assessment process. Analytics for assessment can be added on each question. This is explained in-depth in section 6.3.

2.8.3. Adding nodes in between

Since uAdventure is a fast prototyping tool it is common that conversations change fast and grow during the game development. For this purpose, the lines display a (+) button that allows inserting nodes (Figure 66). when clicked allow adding nodes in between two nodes or even before the conversation root. When clicked, a node is added having as the first child the node in the end of the line. If the node added is a Options node, it will include an empty option referencing to the node by the end of the line.

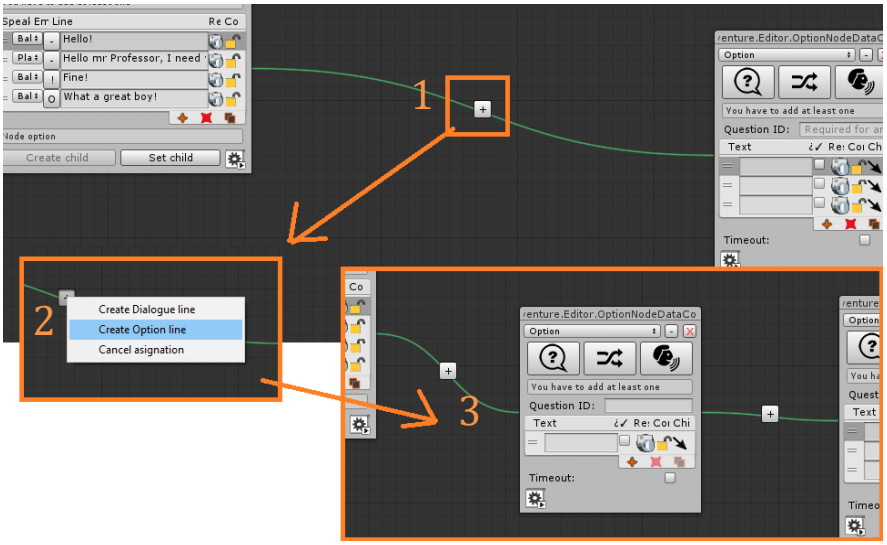


Figure 66. Adding an option node in between other two nodes.

2.8.4. EXAMPLE: Editing a conversation

We can continue with our game story. We just arrived at Balta's office and we must tell him to leave the building as soon as possible.

First, we are going to create a new conversation and click on the "Edit" button in the right panel to open it. On top, we are going to change the name of the conversation to "BuildingEvacuation" and we can add a few dialog lines in our first dialogue node. The conversation with Balta can end in three different outputs: 1) Balta stays calm and leaves, 2) Balta enters panic and leaves, and 3) Balta stays doing some backups. The choices leading to situation one must be clear and calm. In contrast,

situation 2 and 3 may happen for both disagreement and misunderstandings leading to a game over.

We can start with a simple statement (Figure 67):

- Balta: “Hey Pablo, what's that noise?”
- Player: “There’s a fire in the building, we must evacuate all staff immediately.”
- Balta: “And where is the fire?”

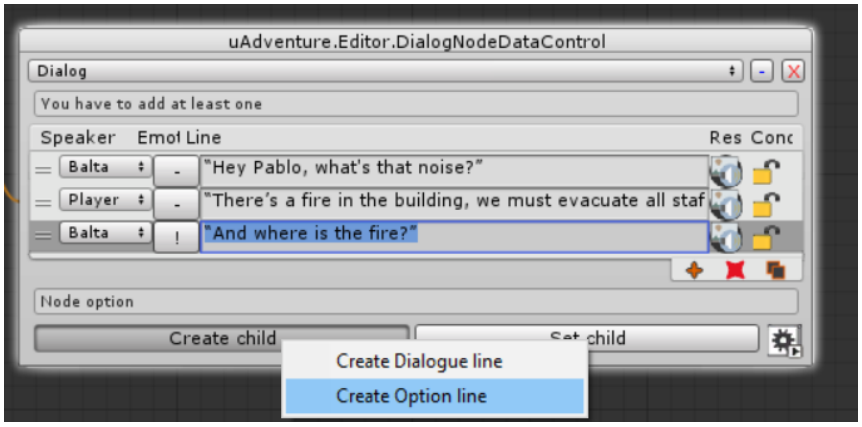


Figure 67. First conversation.

Then we will show three different options that will help us teach some of the educational objectives of this game: “keep calm” and “keep it clear”. For this we will use three options: the first will continue, and so it has to be clear; the second will be unclear, and thus will make Balta ask again; and the last will be too overwhelming and so will make Balta panic.

These three options will follow the next scheme:

- “In office 411”: This is the good option.
- “That doesn’t matter at all”: This option will continue with “You’re not helping me, I can’t evacuate if you don’t tell me where it is.”. To do a loop, when the child is created we have to click on “set child” and click back to the previous options node.
- “In office 411, there’s a huge fire!”: This option will make Balta panic and answer “HELP! RUN!”.

The resulting conversation of the different ways is shown in Figure 68.

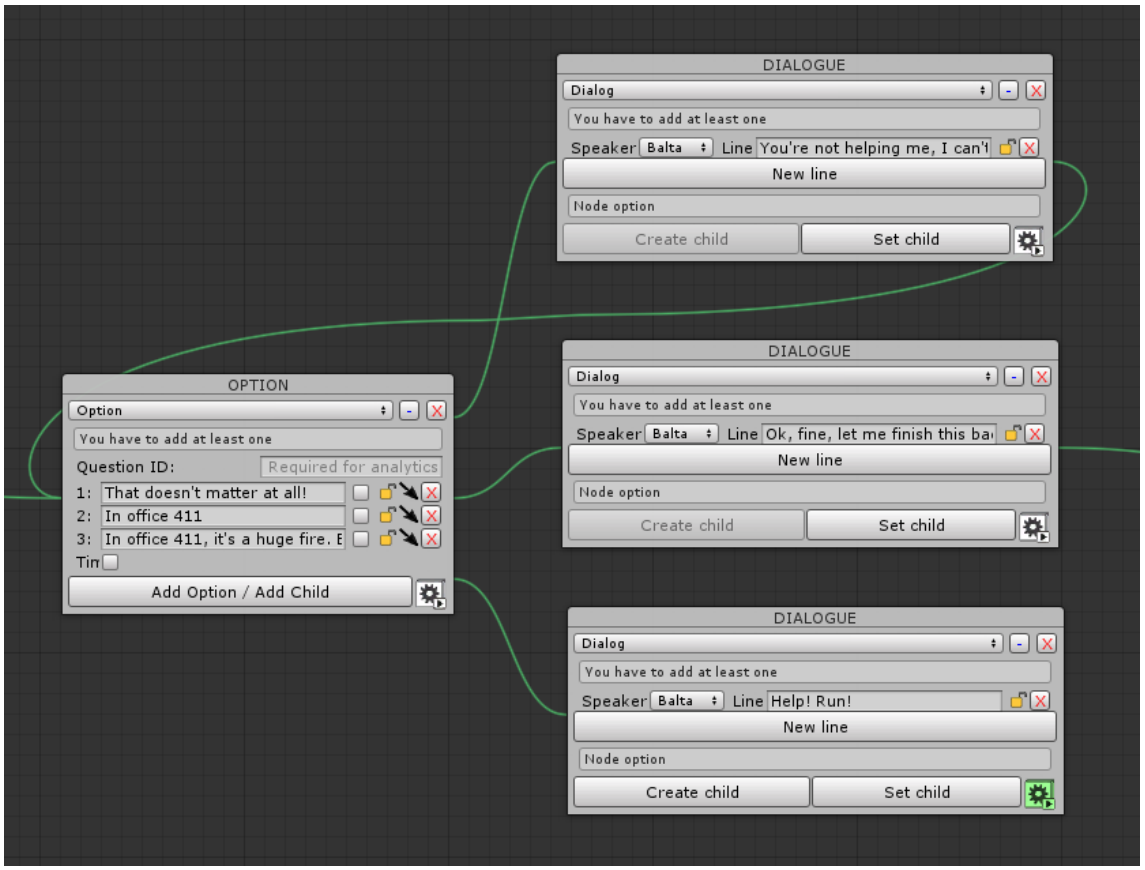


Figure 68. Different conversation paths, the first option loops to the m

When choosing the right way, we can make Balta say, “Ok fine, let me finish this backup and I’ll be downstairs in a minute”. In this case, Balta should leave ASAP, as there is no time for backups during a fire. We can add a few lines to tell Balta to leave or agree for waiting.

- Right way: “No way, you can’t check anything. We should leave now!” (By choosing this option we will change the flags to indicate that the evacuation was done right. This will be later explained in 3.1).
- Wrong way: “Ok, but hurry up”. (By choosing this option we will change the flags to indicate that the teacher stayed and hence we lost the game. This will be later explained in 3.1).

To use different effects depending on the selected node we will have to set them inside of the option dialog, since the effects of the option are always executed, no matter what choice is used (Figure 69).

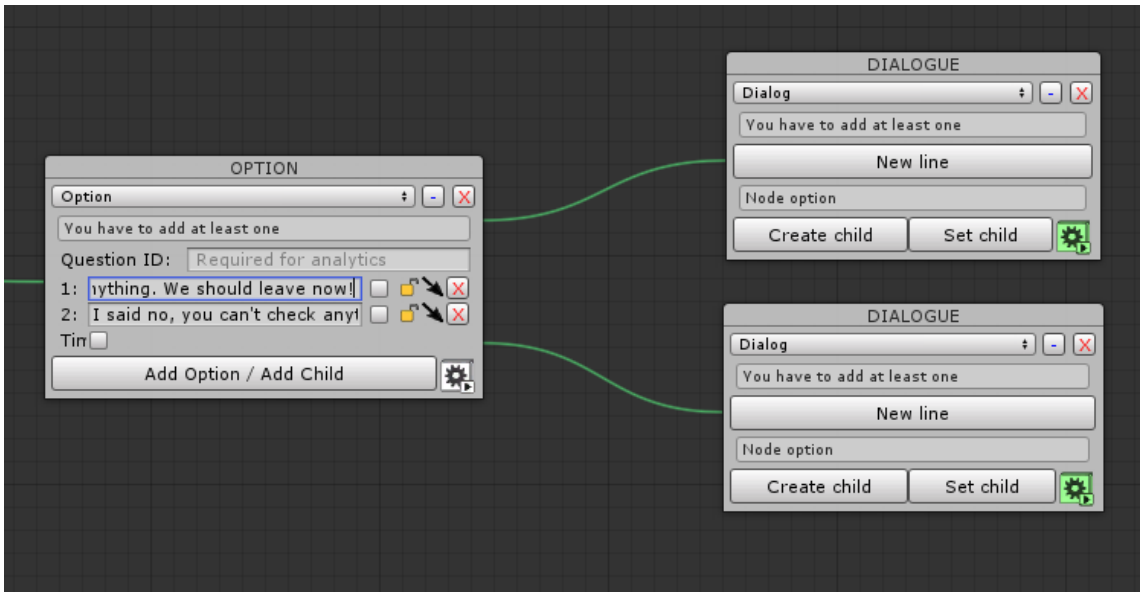


Figure 69. Last option group. The dialogs for each option are left empty, but the effects are used to change the game state.

Now, we can add the conversation to the Balta character by adding a "Talk to…" option to it. First select the "Balta" character in the structure panel and then, go to the "Actions" tab. Click on the add button () and then select "Talk to…" option. A prompt asking us to select the desired conversation will appear with only one option available (Figure 70). Select it and press "OK".

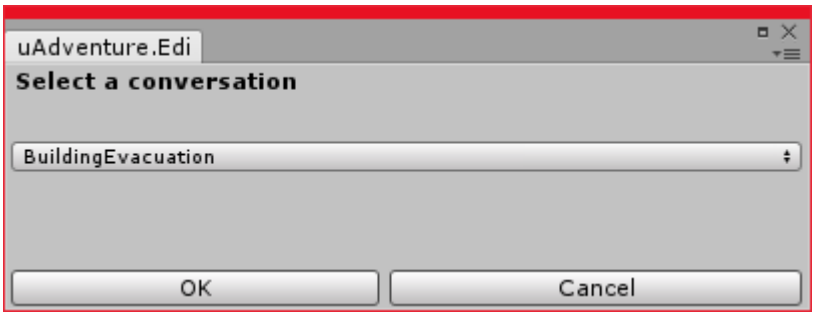


Figure 70. Select Conversation prompt.

The new action will appear first in the list (Figure 71).

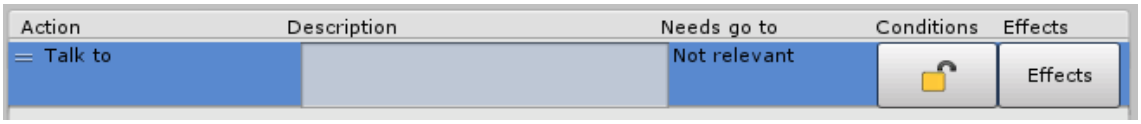


Figure 71. Talk to action added to Balta character.

We do not know yet what effects are. However, by using this talk to action we are using effects in the background, as the only way to play conversation is using a "Trigger conversation effect". For the sake of curiosity, you can click on the "Effects" button and check it yourself. For more information about effects, see section 3.1.10.

Now we can test the conversation. Save the game by clicking on "File > Save" and then press the play button and go to the Balta office. The conversation should go as expected (Figure 72).

Figure 72. Expected dialog sequence.

Later in this manual we will learn how to make the teacher disappear after talking to him, or lose the game if we reach the wrong conversation node.

2.9. The player

The user that is playing the game is generally recognized as the “player”. The interface in the game offers a way for the user to manifest his choices in the game through actions and choices among other interactions. These interactions are represented in game by the player that is a character in the story. Since it is a character, he can participate in dialogs and this is the reason why choices are giving to him to decide which path the story should explore.

The dialog appearance of the player is configured at the “dialog configuration” tab in the player section at the structure panel. This tab is the same as the one used for characters so check section 2.7.2 for more information about it.

On third person games only, the player is represented by an on-screen avatar. The avatar responds to player input and interacts with the different game elements. This avatar animations can be configured as in any other character.

3. Extending basic games: Advanced features

The previous section presented all the necessary tools to create our first basic <u-Adventure> game. However, the platform has some advanced features that increase the potential of games created with it. These features are studied in this section.

3.1. Conditions and effects

The basic elements in <u-Adventure> are items, characters and scenes. However, games require a narrative, a story to be told. In <u-Adventure> the story flow is managed by conditions, which are created based on the values of *flags* and *variables*.

3.1.1. Flags

Flags work as switches, which at any point in the game can be either **active or inactive** (Boolean values of true or false). New flags can be defined and used to create conditions that can be valid or not at a given point, depending on the status of flags.

3.1.2. Variables

As flags can be limited for some situations (such as when something needs to be counted), variables are introduced. Variables are like flags, but instead of being either active or inactive, they can have a value of **0 or more** (integer positive value). By default, variables start the game with the value 0.

3.1.3. Condition editor window

Although the variables and flags are managed in the "Variables and flags window", variables and flags by themselves do not have an implication in the game. Conditions are introduced to give a purpose to a variable or a flag: a condition could be created, for instance, when a flag is active and a variable has a greater value than 3. When all the rules established are true, then the condition is true. To manage conditions, the "Condition editor window" is used.

The condition edition dialog has two main features. First, condition blocks represent the same as an "and" in a sentence. This means that all the condition blocks must be true for the whole condition to be true.

When a new condition block is added, just one line is added to the block including default values on it (by default, the first flag in the list is selected as active). This condition block has a first selector to change between "flag", "variable" and "global state". Once the type of element is selected, then we must select the element that we are going to compare (if there are no elements for the selected type, a warning appears in red). Once we select the type, then the comparison appears on the right side. First, for flags and global states is the same (Figure 73), as it can only be either active or inactive.

Figure 73. Flag condition line set for active.

On the other hand, as variables have a bigger range of values, more comparisons can be done. When adding a variable condition, we need to set the ID of the variable, the comparison function (<, >, =, etc.) and the value (0, 1, 2, etc.) to compare with (Figure 74).

Figure 74. Variable comparisons.

On the right side of the block, we can see two symbols, a "+" button and a "X" button. The first adds a new condition to the block. This means, adding another choice for the condition to be used, just like an "or" in a sentence. Then, this block is considered an "either block" where any of the conditions inside of it make the block true. An "either block" is identified by a darker background surrounding all the conditions that form it. The second button ("X") removes that condition from the block it belongs. If there are only two conditions in an "either block" it becomes automatically a normal condition block as the background reflexes.

For example, we want the "Beef" object can only be grabbed if the "Fish" was grabbed, or either the "Milk" or "Eggs" (or both) where grabbed. Assuming the "beefGrabbed", "fishGrabbed", "milkGrabbed" and "eggsGrabbed" flags are activated when grabbing each of those elements respectively, the condition should be represented as shown in Figure 75.

Figure 75. Condition with two blocks. The first block only includes the "fishGrabbed" active. The second is an either block with either "eggsGrabbed" active or "milkGrabbed" active.

3.1.4. Global states

A more advanced feature in <u-Adventure> is the possibility to define conditions based on "global states" of the games. "**Global states**" are sets of conditions that are defined, allowing them to be reused in different parts of the game (e.g. the game over condition can be a global state Figure 76). These are especially useful in complex games where a point in the game might depend on the combination of different values for several flags and variables.

"**Global states**" are created within their own tab in the "**Advanced features**" section in the left structure panel. "Global states" are displayed and edited just like the conditions of an element, but they are limited to reference themselves.

When editing the conditions of any element, a reference to a "Global state" is added just like the references to variables or flags were added in the previous sections, using the corresponding button.

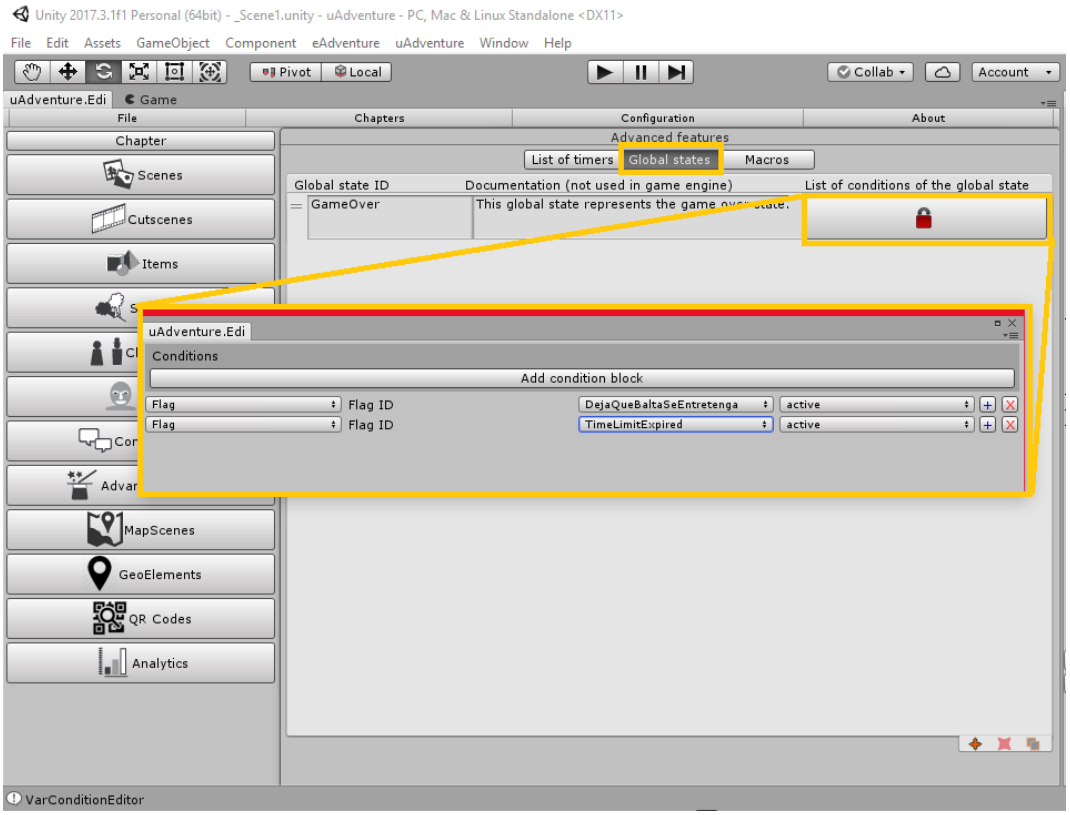


Figure 76. Global state for a game over condition.

3.1.5. Using conditions

Conditions can be used in almost every part of <u-Adventure> games, for example in:

- Actions: to enable or disable interactions with elements
- Conversations: to show or hide new lines and options
- Barriers: to activate or deactivate them
- Exits: to change their behavior, activating one exit or another in the same place
- References to elements: allowing the elements to appear and disappear as needed
- Elements: to change their appearance, by activating or deactivating specific appearances

3.1.6. EXAMPLE: Adding conditions to our phone resources and actions.

In the example 2.4.4 we created a phone with two different views (normal and ringing) and a use action. The idea is to visually identify that the phone has been answered so, after being used, it changes its appearance to the not ringing one. To do this, we are going to use a flag that we are going to call “PhoneAnswered”. When active, the flag will indicate that the phone has been answered so it can change back its appearance to normal. In the next example we will use this flag to block the office exit.

First, we must go to the “Flags and Variables Window”. If you are using the uAdventure layout, this window is already placed in Unity on the right side. Otherwise, you can open it by clicking on the “Chapters > Edit flags and variables” menu. In this window, switch to the “Flags” tab and click on “Add flag” to create a new flag. When asked, write the flag name “PhoneAnswered” and press OK (Figure 77).

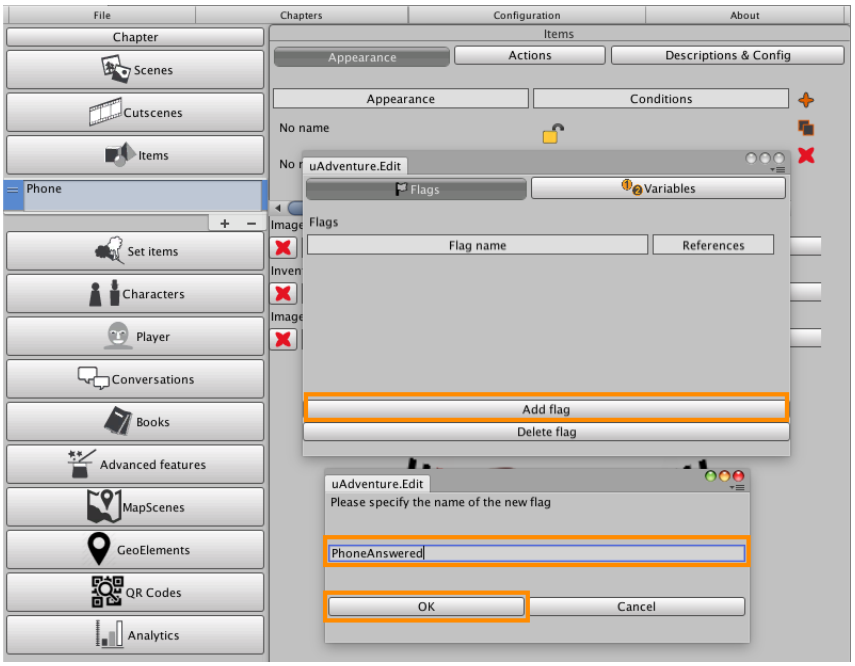


Figure 77. Creation of "PhoneAnswered" flag.

Now we are going to use this flag in the resources of the phone to change them to normal appearance. As mentioned earlier, resources are chosen in order, using the first resource pack that matches its conditions. This way, if we have our first resource pack with a condition, the moment that the condition is not true, it will not be used anymore, leading to the second resource pack.

We can put it in practice by going to the “Phone” item in the structure and switch to the “Appearance” tab. We have two resource packs. To set up conditions to our first resource pack, click on the lock symbol ( ) right to its name. The conditions window will appear as in Figure 78.

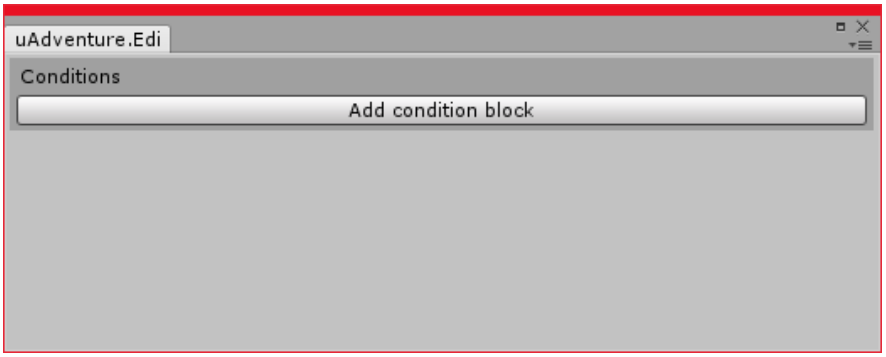


Figure 78. Conditions window.

When clicking on “Add condition block”, you will see a line appear right below with the first block of conditions. This line is a condition filled up with default values (the first values in the list). To configure it, make sure it says “Flag” in the left selector, “PhoneAnswered” is chosen as the desired flag, and “inactive” is selected (Figure 79). This way, when the flag becomes active, the phone will change its appearance to normal.

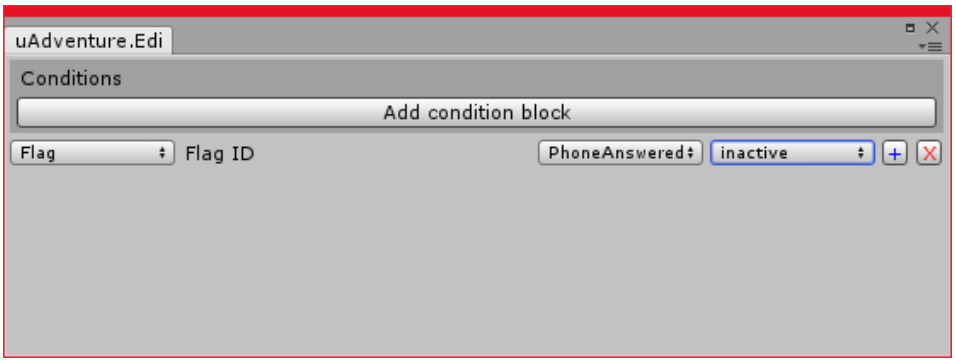


Figure 79. Conditions filled with "PhoneAnswered" flag inactive.

3.1.7. EXAMPLE: Adding conditions to the office exit.

Exits are one of the special cases to use conditions on. By using conditions, we can disable the exit and let the player trapped in the office until he decides to use the phone. We can go to the exit and click on the conditions field to edit them. Inside, we can make the condition match "PhoneAnswered" is active as in Figure 80.

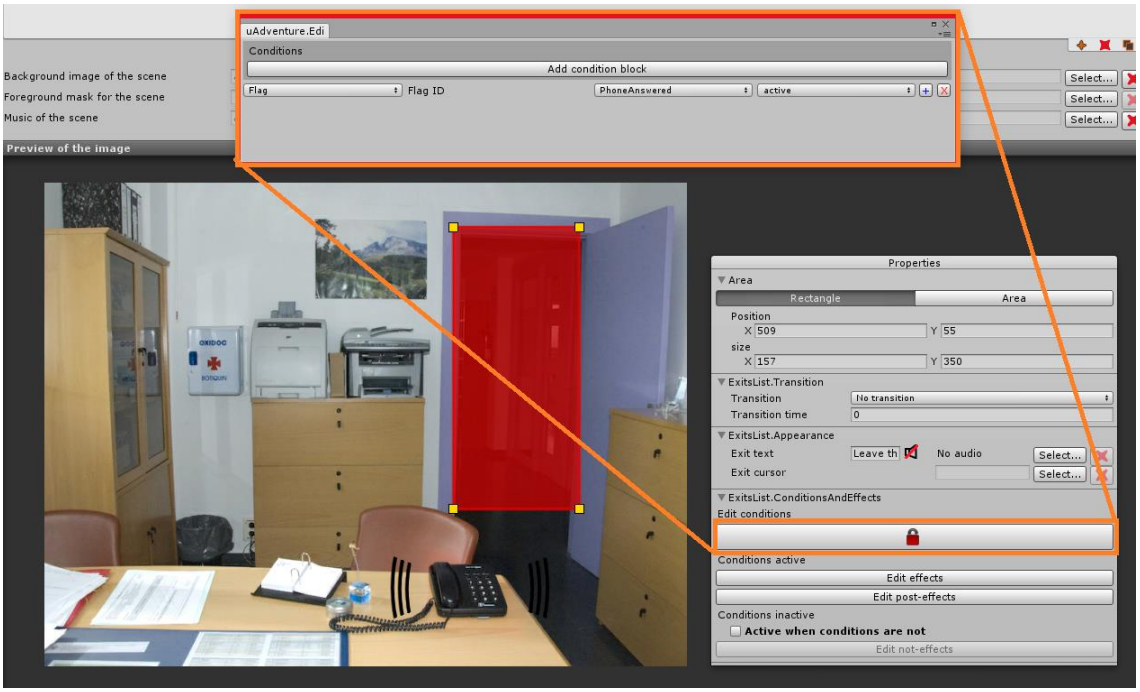


Figure 80. Editing the exit conditions in the element inspector.

In addition, when exits are used, three different types of effects could happen: "Effects", "Post-Effects" and "Not-Effects". The first are the effects that are going to be executed before we go through it. The "Post-Effects" are only used after we arrive into the destination scene. Finally, the "Not-Effects" are executed when the exit cannot be passed.

On this example, we are going to use "not-effects" to show a message letting the player know that he must answer the phone before leaving. To do this, first check the "Active when conditions are not" checkbox and then click on "Edit not-effects" (Figure 81). Then, add a "Speak player effect" saying, "I should answer the phone".

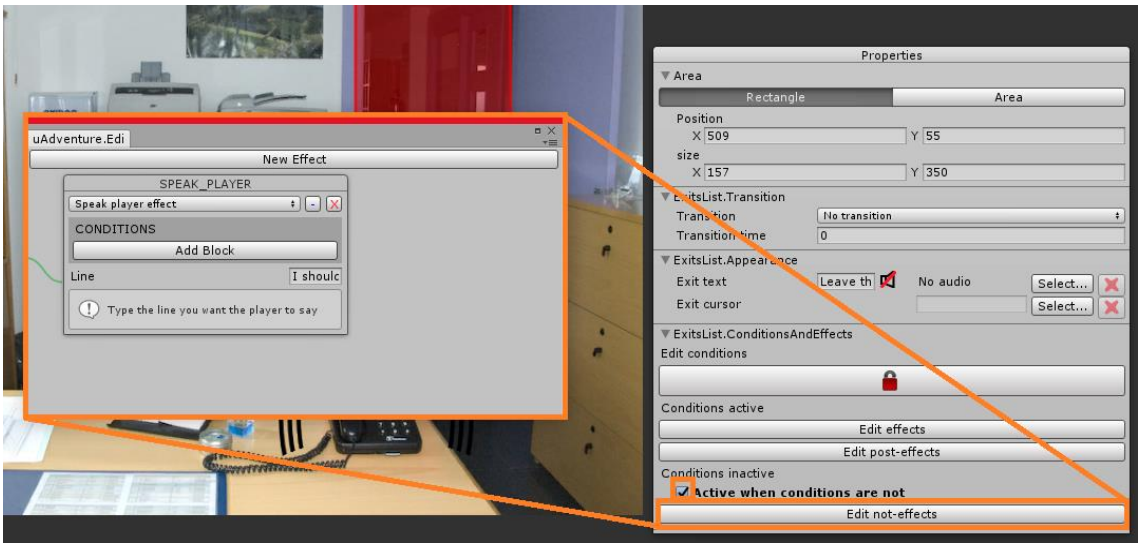


Figure 81. Speak player effect as a not-effect.

Now we can save and run the game to test it. When clicking on the exit we should see the message popping out as in Figure 82.

Figure 82. Result of the not-effect.

3.1.8. Activating and deactivating flags: Effects

To change the state of a flag, <u-Adventure> uses the effect system. Effects are the results of different situations and are defined for specific elements in the game:

- For actions (in items and active areas): when an action is performed by the user, the effects in its block will be triggered one by one.
- In conversations (for each node): every node in the conversation can have an effect block. After all the lines in the node are read, the effects will be triggered one by one, before the next node is started.
- In transitions (for cut-scenes and exits): effects can be added to be triggered just after (post- effects) and, in some cases, just before (pre-effects) the next scene is shown.

Following the same example of the previous sections, if we want to activate the "PhoneAnswered" flag when answering the phone, we must edit the actions of the item. In the "Actions" tab, we should add (or edit) the "Use" action, and in the last column, click on the "effects" button. Once clicked, the effects window will appear. To add a new effect, click on the "New effect"

button on top of the window, and a new node will appear. The node first selector will allow us to change the effect type (in this case, an "Activate" effect as shown in Figure 83). When changing it, we will see how the content of the node will change and the “Flag ID” field appears to select the flag that will be activated.

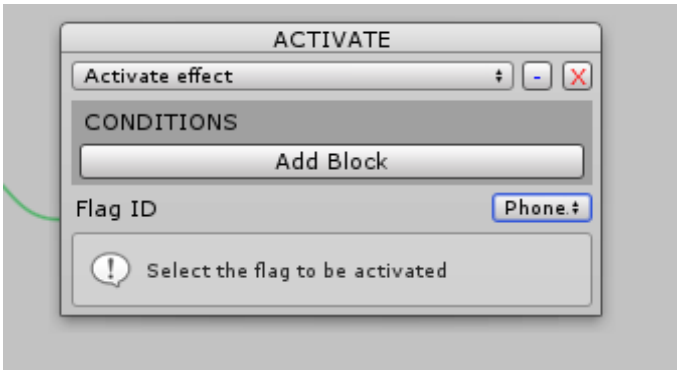


Figure 83. Effect node configured as an activate effect to activate the “PhoneAnswered” flag.

3.1.9. Setting the value of a variable

Just as flags can be activated and deactivated using effects, variables can be assigned a value using three different effects (Figure 84):

- “Set value”: give the variable a pre-defined value.
- “Increment var”: increase the value of the variable by a given amount.
- “Decrement var”: decrease the value of the variable by a given amount.

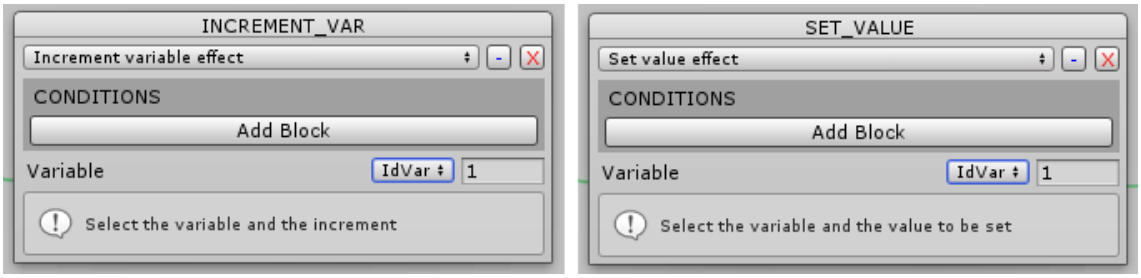


Figure 84. Effects to set the value of a variable or increase its value in 1 units.

3.1.10. Other effects

There are other effects in <u-Adventure> besides the ones used to modify the values of flags and variables (Figure 85).

Effect	Description	Usage
Play a sound	Plays a given sound file (only once)	Enhance attractiveness of games by adding sound to actions (e.g. steps echo when a character moves).
Play an animation	Plays a given animation (set of frames) in the position (x,y) of the scene selected	Enhance visual appearance of games. Can be used, for example, to include elements with special effects such as fades, which are not supported by the game editor.

Player speaks	The main character (i.e. player) says the given dialog line	Useful to provide short feedback messages with little effort.
Character speaks	The character selected says the given dialog line	Useful to provide short messages with little effort.
Show text	Displays the given phrase in a specific point of the scene for a specified time gap.	Like player/character speaks. Main difference is that it allows to select the place where the text will be printed in the scene and the amount of time to be rendered (these parameters are automatically set in other effects).
Trigger a conversation	Starts the selected conversation	Allows to trigger a conversation that is not attached to any character as a "Talk to" action
Trigger a scene	Changes the current scene	Allows to make scene transitions without using exits
Trigger previous scene	Goes back to the previous scene displayed in the game (cut-scenes are not considered). If current scene is the first, an error is prompted.	Useful for implementing menus (using scenes): this effect allows you to define elements that can behave as a "back" button.
Trigger cut-scene	Plays the selected cut-scene.	Allows to trigger cut-scenes without using exits.
Trigger book	Opens the specified book	This is the only way to make a book appear on the screen. Typical usage: define an object to launch the book when it is examined (e.g. an object like a notebook or
Consume an object	Removes an object from the inventory	Allows to make objects disappear from the inventory.
Generate an object	Puts an object in the inventory and removes it from the scene.	Allows to put an object in the player's inventory without defining a "Grab" action
Highlight an object	Highlights the specified object in red/green or blue. It also allows to animate the object. The object will remain highlighted until the scene is changed.	Very useful to guide the player's attention to places of interest at some points of the game.
Move player	Makes the player walk from the current position to a specified destiny on the scene (x,y)	Adds dynamism to the games

Move a character	Makes the character walk from its current position to a specified destiny on the scene (x,y).	Adds dynamism to the games
Move an object (Interpolation)	Interpolates the object from its current position to a specified destiny on the scene (x,y). Object can also be scaled and animated. Interpolation speed is configurable. The object will remain at the destiny position until the scene changes.	Adds dynamism to the games
Launch a macro	Launches the selected macro	Allows for the creation of "set of effects" to be triggered together.
Cancel action	Prevents default effects to be executed in actions	For example, adding a "Cancel action" effect to a "Grab" action will avoid consuming the object from the scene.
Effect with probability	The next effect to be triggered will be chosen randomly between two effects. The probability of each effect can be customized.	Allows for introducing simple random behaviors in the games.
Wait	Blocks the game for a given time gap	Allows to introduce waiting

Figure 85. Table with the effects available in <u-Adventure>.

3.1.11. EXAMPLE: Triggering a cut-scene from an action

Coming from the previous phone example, when we first designed the phone behavior, we wanted to display a conversation that we put in the cut-scene "PhoneConversation". Now that we know how effects work, we can use a "Trigger cut-scene" effect to open our cut-scene.

We can select the phone item and edit again its "Use" effects. Previously, we put an "Activate" effect. Now we are going to add a second effect by clicking on "New effect" and now, we are going to switch its type to "Trigger cut-scene". When selected, we are going to be able to select the "PhoneConversation" cut-scene from the list below (Figure 86).

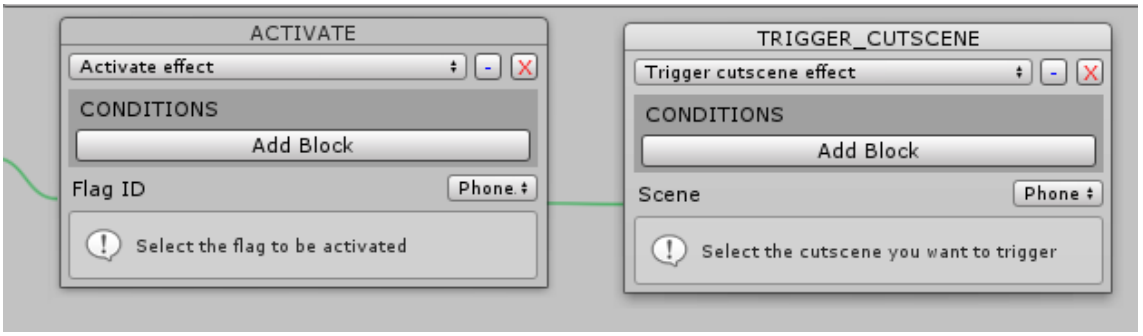


Figure 86. New Trigger cut-scene effect appended after activate effect.

3.1.12. Macros

A *macro* is just a group of effects identified by a unique identifier. This allows launching effect blocks in different parts of the game without duplicating the block.

To add a macro, click on the element **"Advanced features"** in the structure panel, and then select the tab **"Macros"**. Click on the add button (✚) on the bottom-right. After pressing the button, a new macro will be appended to the bottom of the list with an automatically generated ID. This ID can be changed as well as the documentation. To edit the effects of the macro just click on the "Effects" button and the effects editor window will be opened. The only restriction in macros is that a macro cannot contain an effect "Launch macro" to itself.

To launch all the effects in a macro, just add a "Launch Macro" effect at any point of the game.

3.1.13.Variables and flags quick menu.

In the conversation window and in the effects window, the variables and flags be easily accessed on the top left corner as a dropdown menu.

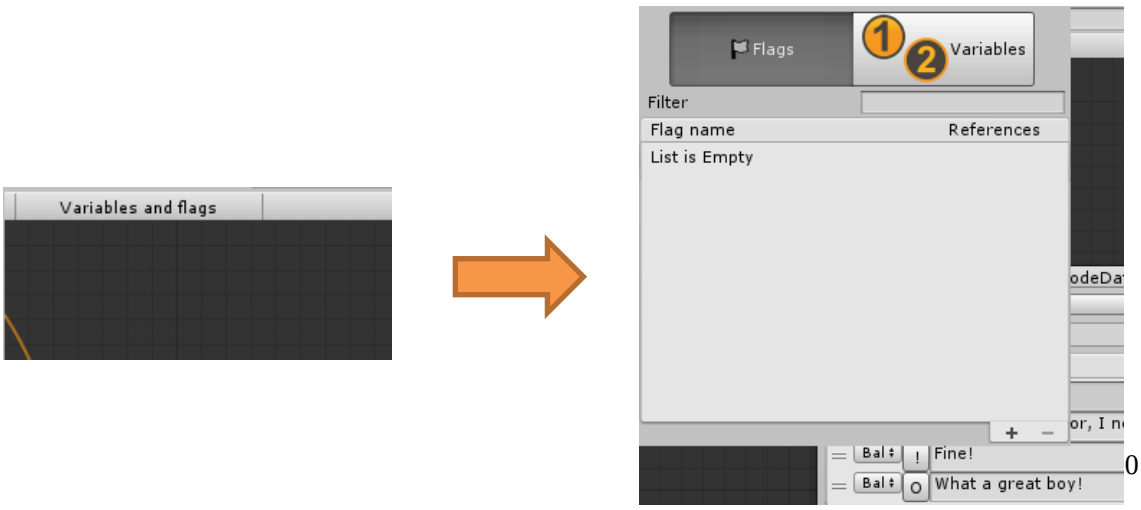


Figure 87. Variables and flags quick menu.

3.2. Organization of the element references in the scene

This Section covers different features provided in the "Element references" tab of the scenes. This tab is divided in two vertical sections: a reference list (top) and a preview panel (bottom). Sections 3.2.1, 3.2.2, 3.2.3, cover settings of the reference list. Section 3.2.4 addresses the preview panel, and Section 3.2.5 the element inspector.

3.2.1. Layers

The order in which elements referenced in a scene (characters, items and set items) are drawn can be configured. This feature is important to represent depth (z coordinate) in 2D scenes like those present in <u-Adventure> games.

A layer is the order in which an element will be drawn in the scene. Element in layer 0 is the first to be drawn, and therefore will be perceived as the farthest from the observer (player).

To change the layer of each element reference, go to the scene and click on the tab "**Element references**". The layer of each element is shown in the first column of the list of references. You can alter the layer of an element by dragging it up and down in the list (see Figure 88).

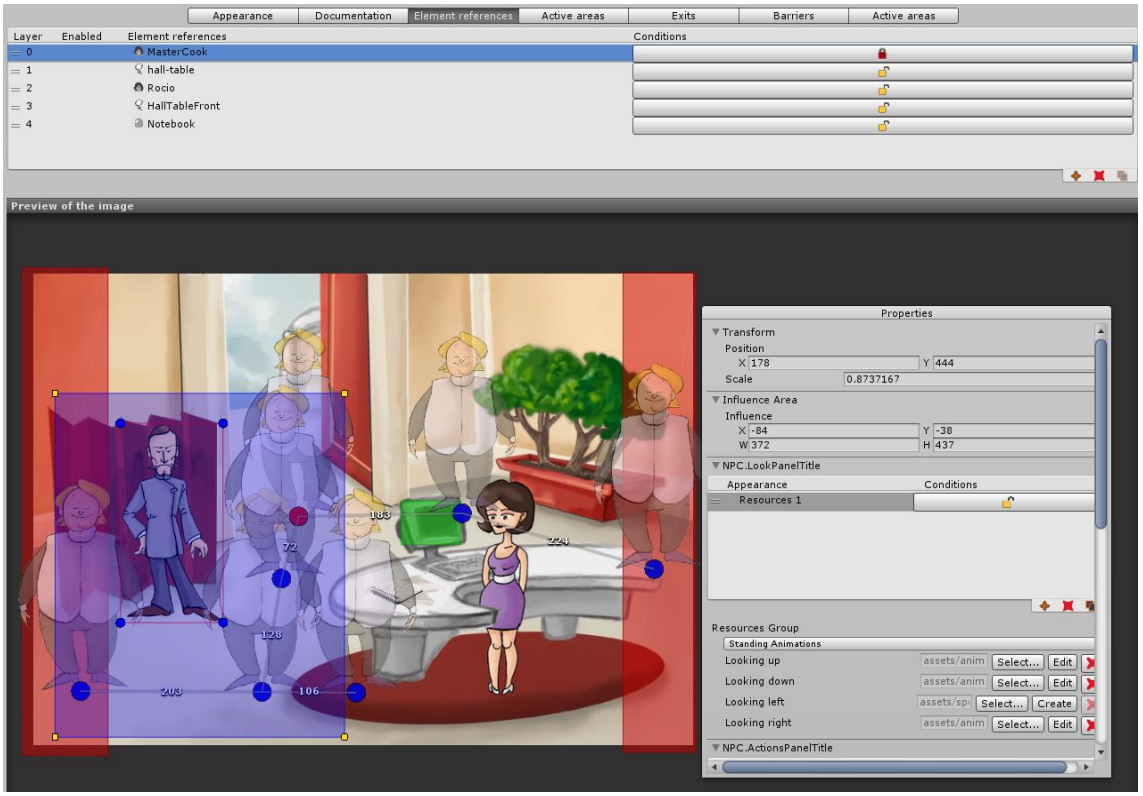


Figure 88. Edition of the layer of element references.

The layer of the player is treated differently. Since the player can move in the scene, the <u-Adventure> engine will assign its layer automatically depending on how far is estimated to be from the observer according to its current position and trajectory. To determine whether the player or an element must be drawn closer to the observer, coordinates y of both are compared (the lower y is in the front).

For example, this allows the player to get drawn before an element when it is supposed to be farther. In Figure 89 we can see that when the player enters the scene it is behind the table, but after

moving towards the bottom of the scene the player is drawn in front.

Figure 89. Example of using layers to adjust depth in the scene.

3.2.2. Conditions

When an element reference is selected in the "Element references" tab (either using the preview panel on the bottom or the reference list on the top) there is a button with a lock symbol ( ) that allows the edition of conditions for that reference. This allows setting restrictions on the visibility of each element reference by using flags, variables or game states (as explained in section 3.1). Elements can appear or disappear in the scene depending on the conditions established.

3.2.3. References list

In this panel all the element references and the player are displayed. They are ordered according to their layer. References can be added or deleted using add ( ) and delete ( ) buttons on the bottom-right corner of this panel. Also, when a reference is selected it is possible to duplicate it ( ). When pressing the add button, a selector for the available element types appears. When any type is selected, a pop-up lets the user select any of the previously-created elements. For more information on how to create each type of element see sections 2.4.1 (items), 2.5.1 (set items) and 2.7.1 (characters).

In addition, by dragging the element in the list using the left handle it is possible to modify the layer of any element reference.

3.2.4. Elements preview

This sub-panel provides a preview of how the elements will look in the scene. In addition, this panel allows modifying the position and scale of each element reference. To modify an element reference, just click on the element and then drag any of its blue round corners to change its size (scale), or drag the element itself (from the middle, for instance) to place the element in a different position.

3.2.5. Element inspector

In addition to the preview with the different handles, when an element is selected the inspector appears in the bottom right corner of the scene preview. This inspector allows to examine all the properties of the element. In the case of element references, those properties include both positioning information and all the different properties already editable in the element section.

In fact, all the different tabs present in the element section will appear as collapsible sections in the element inspector (i.e. an item will have its three tabs including "Appearance", "Actions" and "Documentation" and the extra "Transform" to manage the reference position and scale). The only

difference between this editor and the ones in the element section is the absence of the preview.

It is important to mention that changing any of the properties in the element inspector will affect all the element instances in other scenes, except for the transform values (including the position and the scale) that are managed separately in any scene.

Finally, the element inspector can be dragged and moved to any of the corners in the elements preview area by clicking it on its top.

3.3. Active areas

Active areas are rectangles or polygons defined as part of the scene that define a part of the background that can be interacted with. Active areas are especially relevant to create interaction with parts of the scene in photo-realistic games as they avoid having to create items or characters and adjusting them to the photo.

An active area behaves much like an item, but it belongs to the scene and it is not created independently (Figure 90). They are defined in the same way as exits, and can be of different sizes and in different positions.

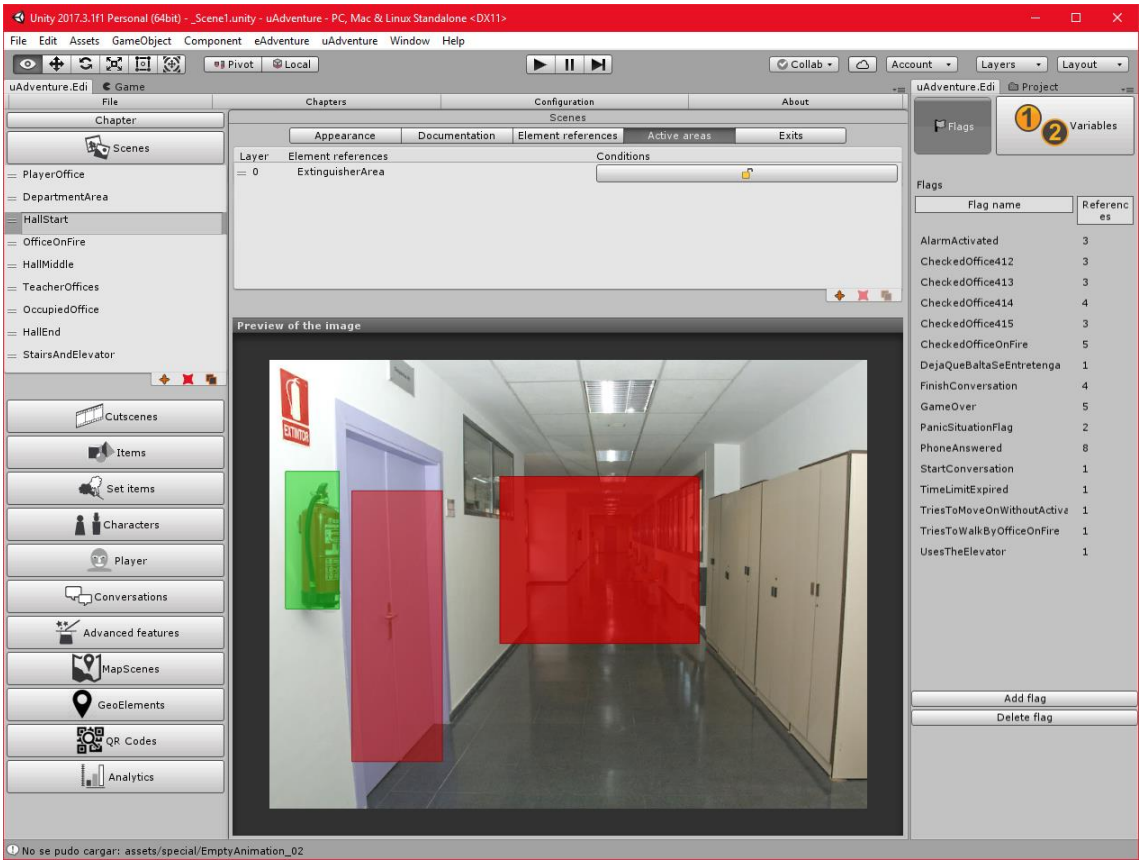


Figure 90. "Active areas" tab in the scene edition panel. Active Areas ca be identified in green while exits are red.

3.3.1. EXAMPLE: Adding an active area with actions.

In this example, we are going to add a fire alarm in the second hall on top of the fire alarm. When pressed, a conversation will say that the alarm has been activated, and if we interact with it again, the alarm will say that it is already active. Also, we are going to make the exit towards the offices inaccessible until we activate the fire alarm, and show a text back to the player inviting him to activate the alarm.

First, go to the “HallMiddle” scene in the structure panel and switch to the “Active Areas” tab. Then, click on the add button to add a new active area in the scene. As you can see, a new green rectangle appears in the middle of the scene. That is our new active area that we can move and resize until it fits the fire alarm. The result should be as in Figure 91.

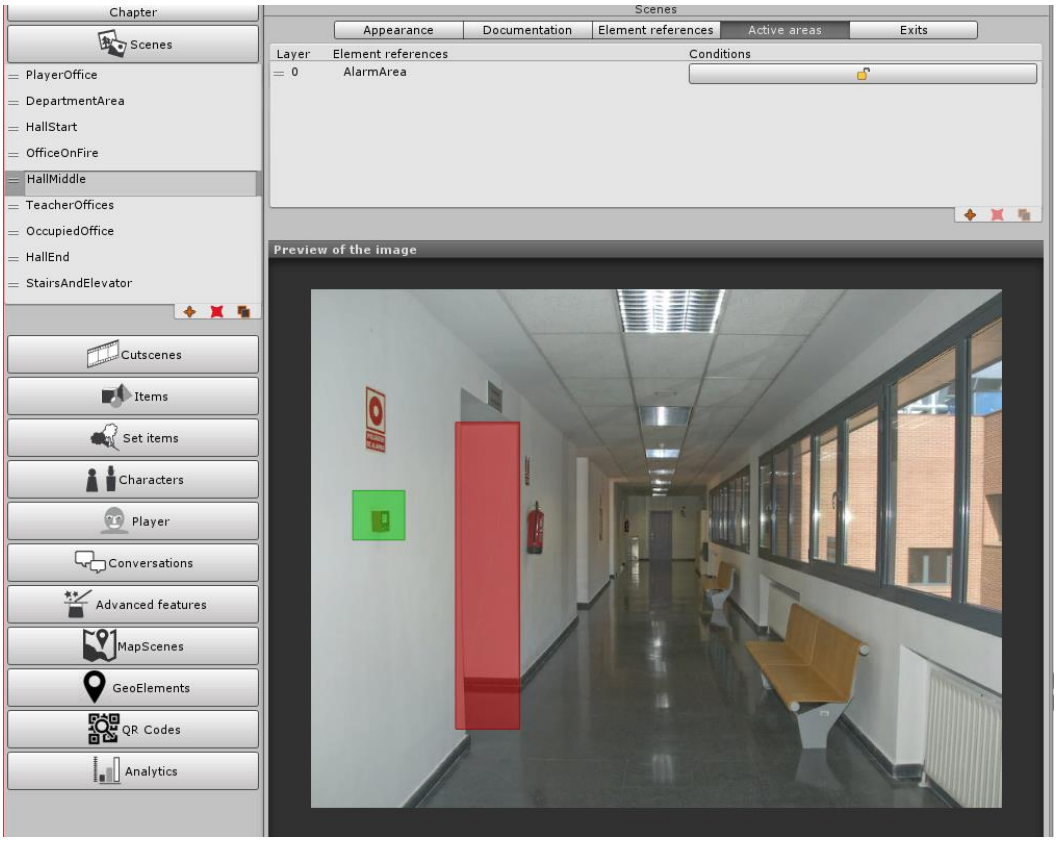


Figure 91. Fire alarm active area.

Now that we have our fire alarm active area settled, we can set the behavior when clicked. We said that a condition related with activating the alarm will change the text of using the alarm. That means we are going to use a flag to control if the alarm has been activated. Go to the “Flags and Variables window” and add a new flag named “AlarmActivated”.

We are ready to create actions in our active area, so go back to the scene and select it to be able to see its contents in the element inspector (on the inspector you might need to scroll a little bit to find the actions list). Click on the add button to **add a new use action**, and then, open the effects by clicking on the “**Effects**” button.

Inside of it, we will need two effects. First, we are going to activate the flag “AlarmActivated”. Second, we are going to show a text saying that the alarm has been activated. The effect sequence should look as in Figure 92.

At this point, we could set conditions to the effects inside the actions to change the text that is shown when performing the action. Alternatively, there is a different approach for changing an action behavior. When multiple actions of the same type are present in an element the first one which conditions are matched is used. Hence, by adding another action with the same type and setting up the conditions of each one, we will be able to switch from one action to the new one. To do this, on the conditions tab of the first action, we can add the condition of “AlarmActivated” is inactive. This will disable it when clicked. In our second use action we are going to add an “speak

player effect" saying that the alarm has already been activated.

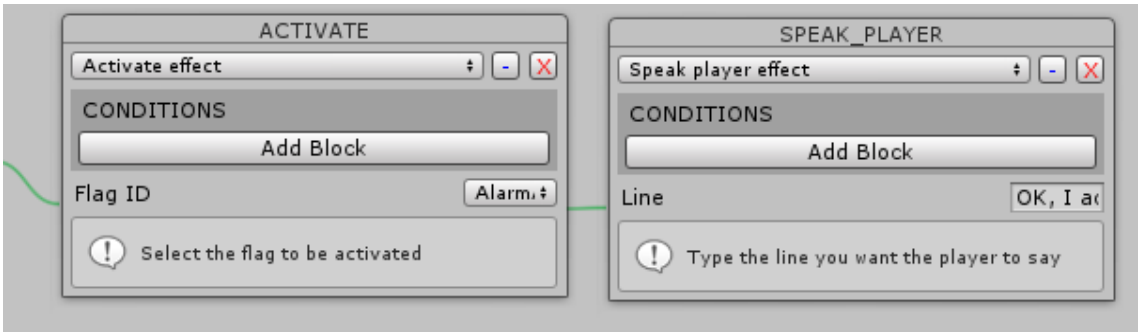


Figure 92. Activate alarm effects sequence.

Now, following the same indications as in section 3.1.7, change the exit towards the offices to be only accessible when the flag "AlarmActivated" is enabled.

3.4. Barriers

Barriers (only available in third person games), like exits or active areas, are rectangular areas or polygons defined on a scene. Barriers provide the ability to limit the players movement (other characters in the game are not constrained by barriers). Conditions can be defined as to when the barrier is active and when it is not. When a barrier is active, the player will stop just before it until the conditions are not met anymore. These elements are edited just like any other, in the scene edition panel, in the "Barriers" tab (Figure 93).

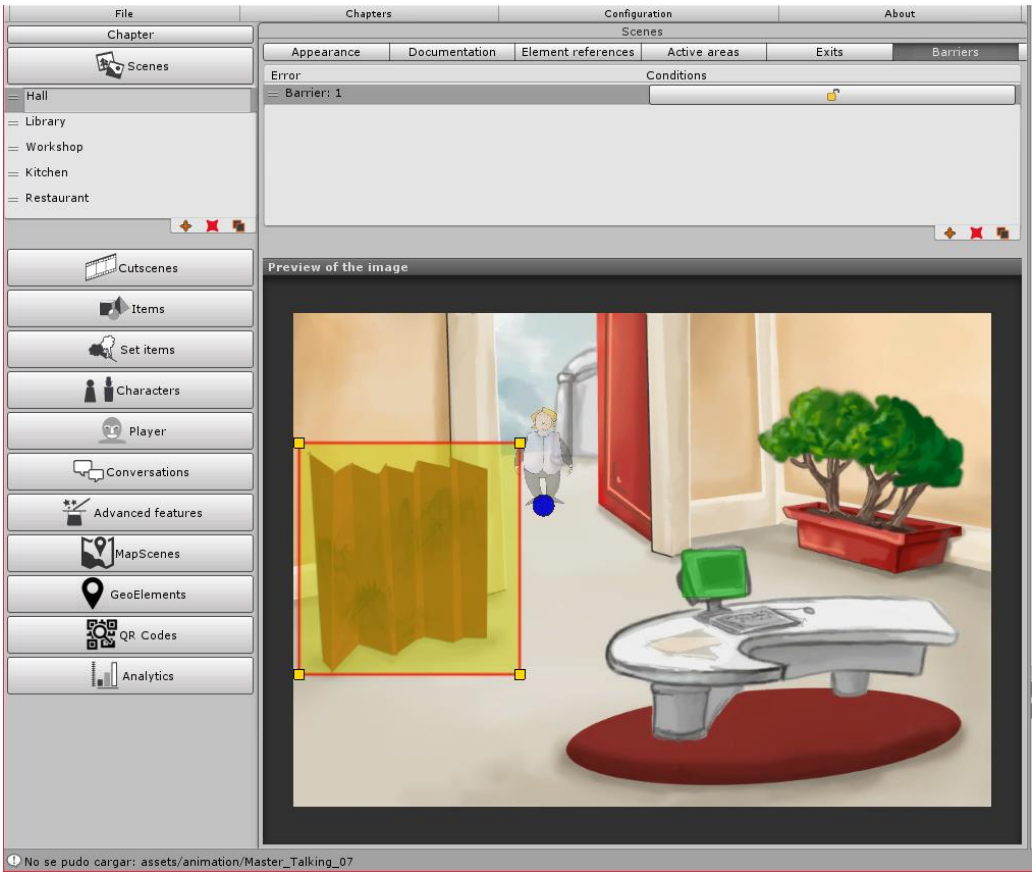


Figure 93. "Barrier" edition tab, in the scene edition panel.

3.5. Player movement

The way the player moves in a scene can be configured in three different ways: hiding the player, setting an initial position or using trajectories (only for third person games). When the "Hide player" option is selected, the player is not shown in the scene, and hence, the scene behaves as a first-person game. When an initial position is used, the player will only be able to move along the horizontal line that goes through that point. Besides, the player will start (by default) in that position. No further configuration is reached and all objects can be reached if near their X coordinate.

The other option, using trajectories, requires the creator to explicitly draw all the paths for the player. This is done by first defining the nodes or points of interest along these lines and the lines from one to another. There is no limit in the number of nodes, lines or loops, but it is strongly advised that they are kept to a minimum for efficiency.

When editing the trajectory of a scene, there are several tools at our disposal (Figure 94). These tools are over the preview of the scene. They are (in order):

- Edit nodes: this tool is used to select, move and scale nodes, as well as creating new ones.
- Edit sides: this tool is used to add new sides (paths from one node to another) to the trajectory. Sides are created clicking one node and then another.
- Select initial node: this tool allows for the selection of the initial node in the trajectory. This is the node where the player will appear in the scene.

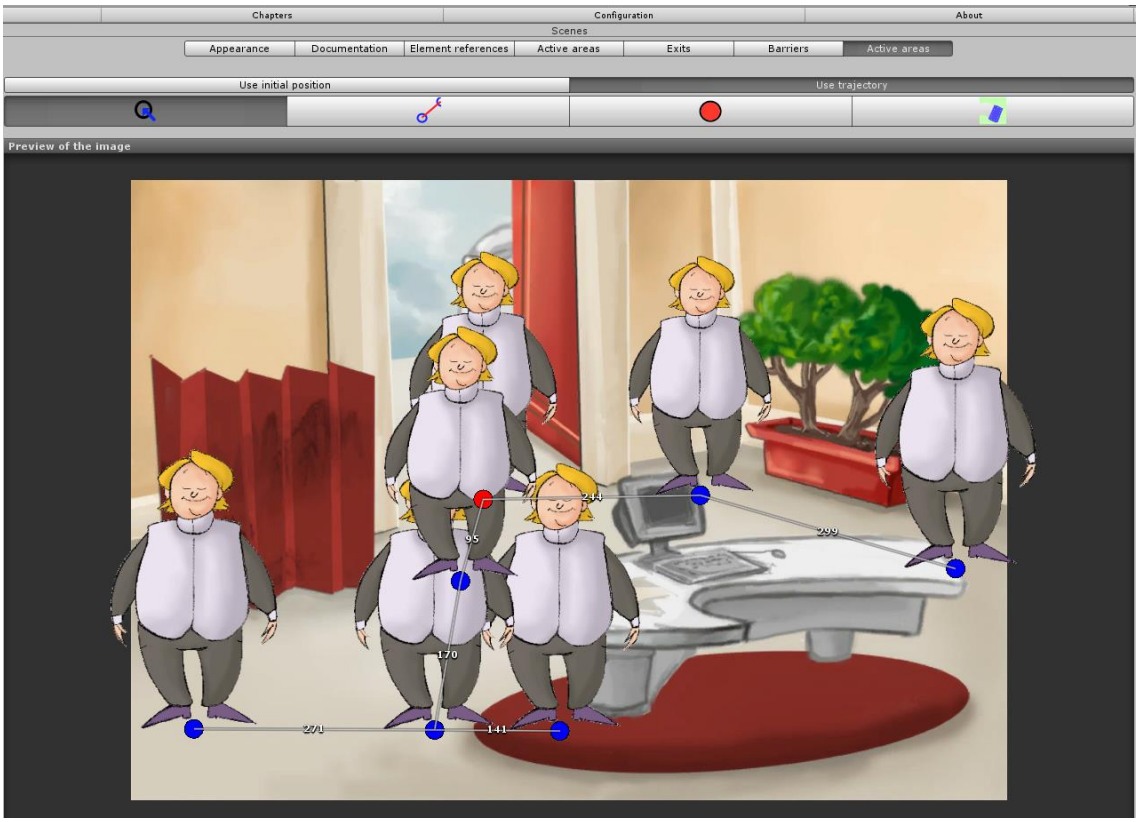


Figure 94. "Player movement" tab in the scene edition panel with trajectory mode enabled.

- Delete: this tool is used to delete any node or side by clicking on it.

The scale of the nodes is used to create a sense of depth in the scene, making the player seem bigger or smaller depending on the current node. If there is a path connecting two nodes with different scales, the player will be draw with a scale equaled to the pondered mean at every point. The speed with which the player travels along these sides can also be modified by changing the length of the side (the number that appears over the line).

Barriers only affect the trajectory if they directly cut a path. Objects can only be interacted with if their influence area intersects the path. The influence area is edited just like other scene elements. Influence areas are also applied to exits and active areas. Influence areas can only be edited while the element is selected in the "Element references" tab (Figure 95). Active areas appear in blue.

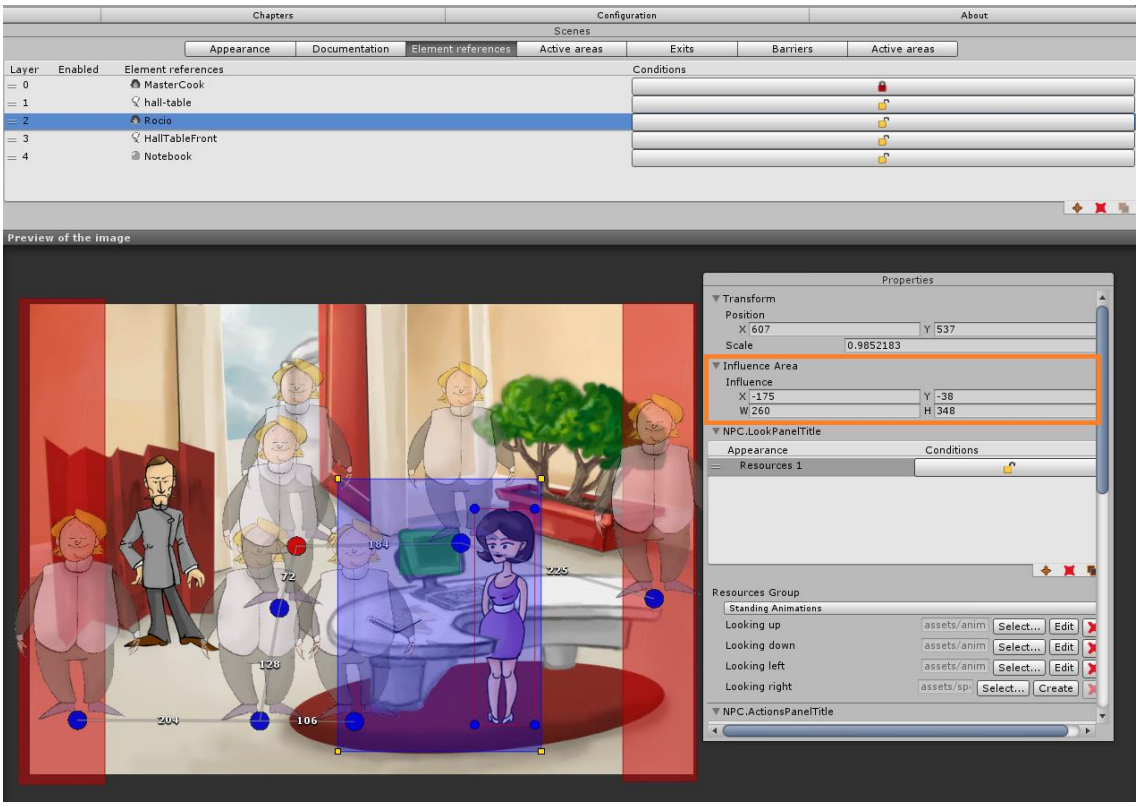


Figure 95. Editing the influence area of an element reference in the "Element references" tab.

Additionally, influence areas can be edited in the element inspector too, as their properties appear in the second position. The influence area property is not shared with the other same element references, so its value has to be configured for each scene as the element appears.

3.6. Timers

Timers allow triggering blocks of effects periodically or after a specific amount of time has elapsed.

To configure timers, click on **"Advanced features"** on the structure panel. The first tab displayed on the right panel will be the timer list (Figure 96). Timers can be added or deleted using add (✚) and delete (✖) buttons on the right side of the timer list (which by default is empty).

The basic idea of how a timer works is simple. When the defined set of conditions are met, the timer will start counting until a time limit is reached. Then, a block of effects will be triggered. Nonetheless, timers can be configured to tweak this behavior in several ways.

First, timers can be forced to stop by defining a set of "End conditions". When these conditions are met then the timer will abort the count (if the time limit has not been reached yet). In addition, another set of effects can be triggered when the timer is forced to stop. Second, it can be configured if the timer should remain active during the whole game or just once, establishing conditions for the timer to start again. This behavior allows reusing the timer in different parts of the game, or running in loops (the timer will restart if the time limit is reached, until end conditions are met).

The full list of parameters that can be configured for a timer is as follows:

- **Time** (second column of timer list): this is the time limit. When reached, the timer will trigger the effects block and restart or end, depending on the configuration.
- **Display in game** (third column of timer list): if active, the elapsed time will be rendered

on the upper right corner of the screen. Other parameters will be available then on the panel below the list:

- Display name: Text that will be displayed along with the elapsed time.
- Count-down: If active, the timer will count backwards and trigger the effects when zero is reached.
- Show when stopped: If active, the text and time elapsed will be displayed on the screen even if it has not started counting or if it has been stopped.

- **Documentation**: Text for internal documentation, not used in the games.
- **Loop control**:
  - Multiple starts: defines if the timer to start once or multiple times.
  - Runs in loops: defines if the timer has to start again after reaching the time limit.
- **Conditions to start the timer:** the timer will start counting when these conditions are met.
- **Conditions to stop the timer:** the timer will stop (even if the time limit has not been reached) when these conditions are met.

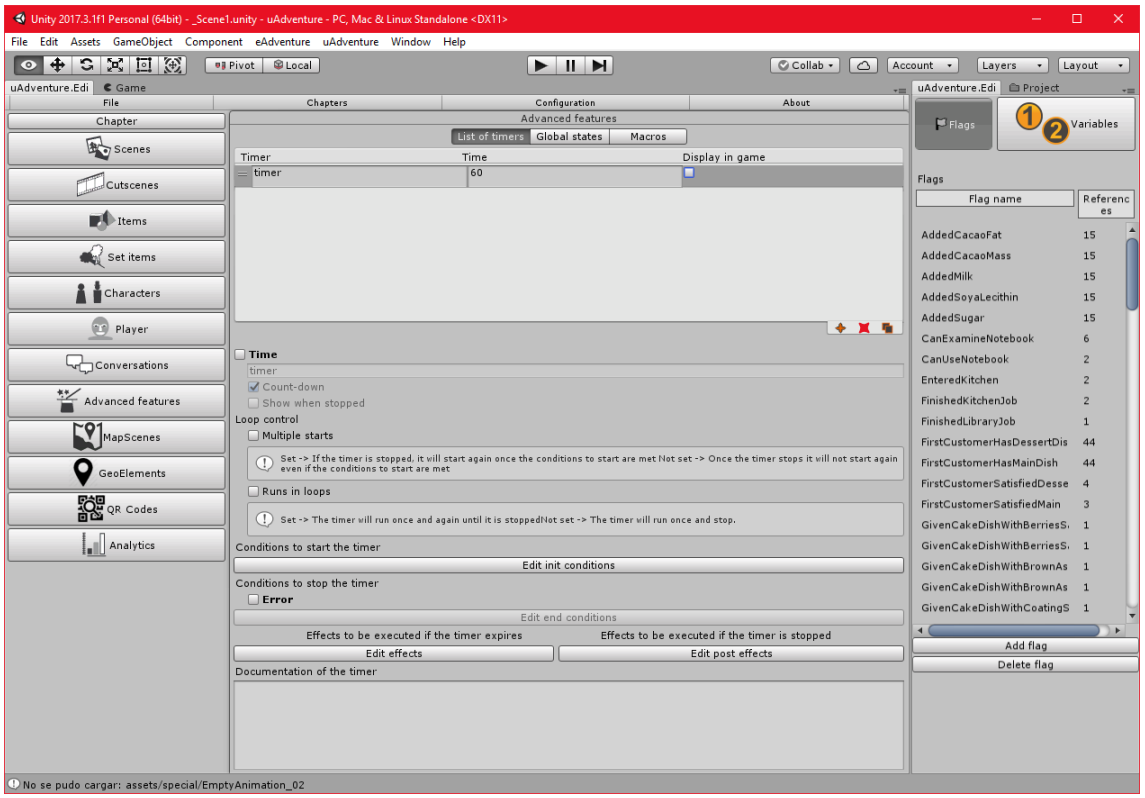


Figure 96. Edition of advanced features, "Timers" tab.

3.7. Custom actions

In section 2.4.2 actions were introduced. Those were predefined actions, which have a specific default behavior. For example, a "Grab" action is always going to remove an object from the scene and add it to the inventory.

Custom actions allow defining interactions in the game with no specific default behavior, extending the expressiveness of the <u-Adventure> platform to suit any specific needs.

There are two types of custom actions depending on the number of game elements involved. If the action is unary (the action is executed over a single element), like "eating", "throwing" or "turning on" it will be called simply an **action**. If the action is binary (the action uses the first element but is executed on a second target element), like "combining element1 with element2" or "giving element1 to character2", it will be considered an **interaction**.

The drawback is that custom actions require setting the images for the buttons that will use the player to trigger the action. For default actions <u-Adventure> sets predefined buttons (e.g. a "hand button" for use, grab or use-with actions), although these will also be customizable. However, as the platform cannot foresee the meaning of custom actions the author of the game will compulsorily have to define them.

In addition, it may also be required to define a custom animation of the player executing the action. For default actions <u-Adventure> uses animations defined for the player, but these may be inaccurate for custom actions. However, defining the player animation is optional: if not set, player's "use animation" will be applied. In first person games, this step is not required.

Two images are required for an action button. The first will be used to represent the button in normal state. The second will be used to represent the button when the mouse is over (Figure 97. Example of normal and over images for a custom action button.Figure 97).

Figure 97. Example of normal and over images for a custom action button.

To add a custom action, click on the add button () on the right side of the action list (actions tab). Within the contextual menu that will appear, option "Custom action" must be selected.

Then the editor will ask for a name. Use the dialog to introduce a representative name for the action (take into account that this will be the text used in the game for this action). After clicking on the "OK" button the system will ask if the custom action is unary (action) or binary (interaction).

If interaction is selected, the system will also ask which is the target element involved in the action.

A new custom action will be added to the list. The name and target element (in the case of interactions) can be edited, although it is not possible to change the type of the action. Below the action list an edition panel will appear. This panel contains two different tabs: **"Personalization"** and **"Configuration".** Use the "Personalization" tab to select the images for the action button and animations of the player (optional). Use the "Configuration" tab to edit the effects of the action (see section 2.4.2 for more details).

NOTE: Please take into account that the optimum resolution for action button's images is 40x40 pixels. Recommended format is PNG. Buttons defined with images that do not comply with these guidelines may behave unexpectedly.

3.8. Built-in art resources edition tools

Although art resources have to be produced with specialized tools (e.g. Adobe Photoshop™), <u-Adventure> provides support to make some slight adjustments or create simple art resources. In this

section we can find the resource editing tools. Previously, in <e-Adventure> more editors such as the image background editor and the HTML editor were available. For now, <u-Adventure> only has the animations editor, but more editors and Unity tools will appear in future releases.

3.8.1. Animations editor

This editor allows customizing animations for the <u-Adventure> game engine. Animations produced with this editor will be stored in format ".eaa.xml", which is an evolution of the previous ".eaa" proprietary format for <u-Adventure>, adapted to be compatible with Unity. When importing an old <e-Adventure> project, all the ".eaa" files are automatically converted to ".eaa.xml". These animations will be used either for cut-scenes and characters. The animation editor also converts old <e-Adventure> format (groups of frame images with a common file name *_NN.png / *_NN.jpg where NN is the index of the gram) to the ".eaa.xml" format automatically. To make that conversion, just click on the "Edit" button of the animation field you want to upgrade.

From any animation field, when clicking "Create/Edit" the animation editor will appear (Figure 98).

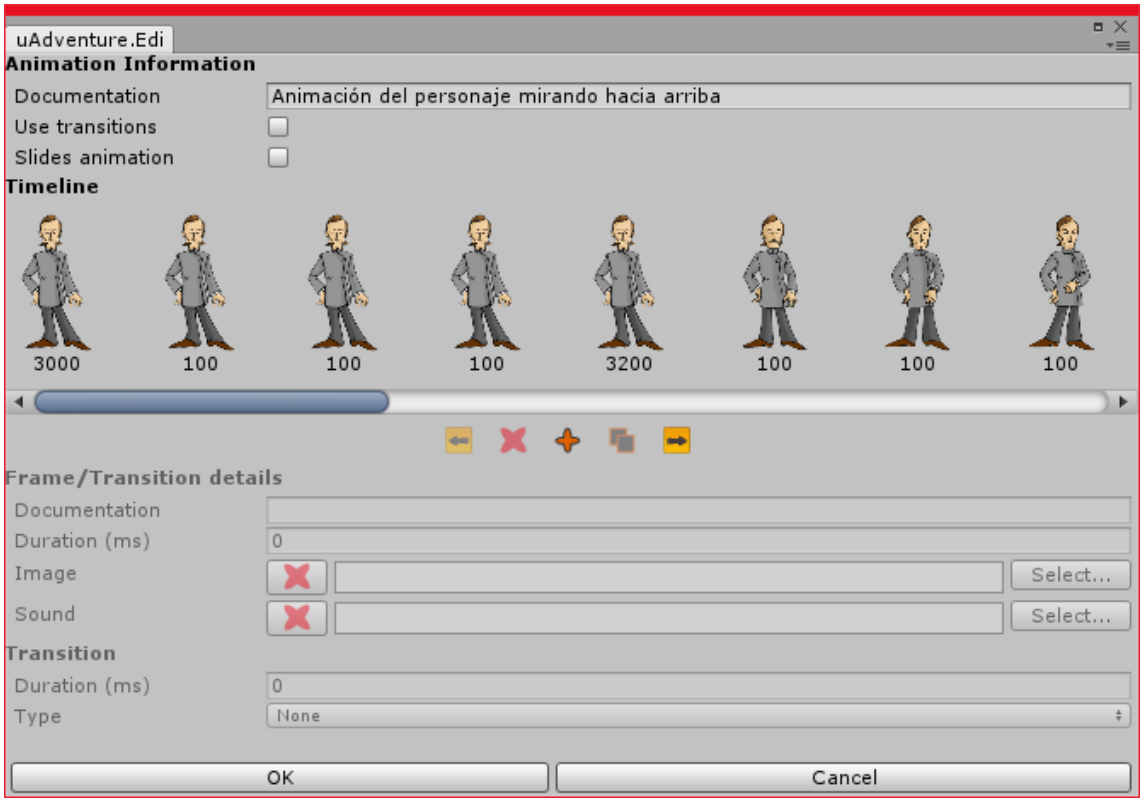


Figure 98. Animation editor.

The documentation field is used only to provide an internal description of the animation, but is not used in the game. Below this field, the next fields are presented:

- *"Use transitions"*: activates/deactivates the possibility of including transitions between frames.
- *"Slides animation"*: activates a special behavior for slides, adding the possibility to block the next frame until the player clicks on the screen. It allows players to read slides at their own pace.

The timeline is presented under these options. In this timeline, frames and transitions are displayed in order. Frames can be added, removed or duplicated using the controls below ().

In addition, below each frame or transition its duration (in milliseconds) will be displayed.

Frames and transitions can be selected by clicking on them in the timeline. When a frame is selected its properties can be edited using the panel that appears just below (titled "Frame/transition" details).

When a frame is selected the next properties can be edited (Figure 99):

- Duration (ms): Time that the frame will be displayed on the screen (in milliseconds). By default, this value is set to 40.
- Image: The image of the frame. Any png, jpeg or bmp image is accepted.
- Sound: An mp3 or midi sound that will be played synchronized with the frame. This way, we can add sound effects to animations (e.g. step sounds when a character walks). Please take into account that these sounds will not stop when the next frame is loaded, so it is recommended that the frame's duration is set to the length of the sound file selected.

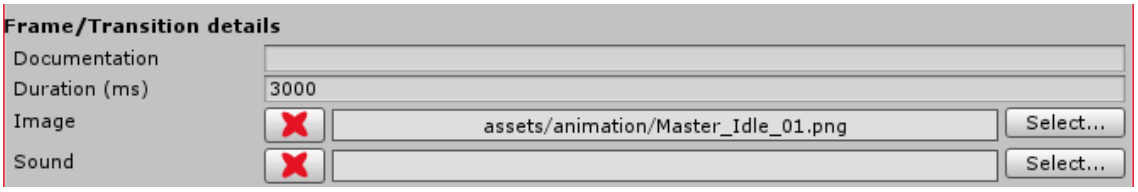


Figure 99. Properties of a frame in the animations editor.

When a transition is selected, the next properties can be edited (Figure 100):

- Duration (ms): Time the transition will be displayed (ms). By default, this is set to zero.
- Type of transition: three types of transitions are currently supported:
  - *Fade in*: fades the previous and next frames. The previous frame disappears and the next one appears progressively.
  - *Horizontal*: the previous frame moves towards the bottom of the screen and the next frame appears on the top.
  - *Vertical*: similar to horizontal, but in this case, movement is from left to right.

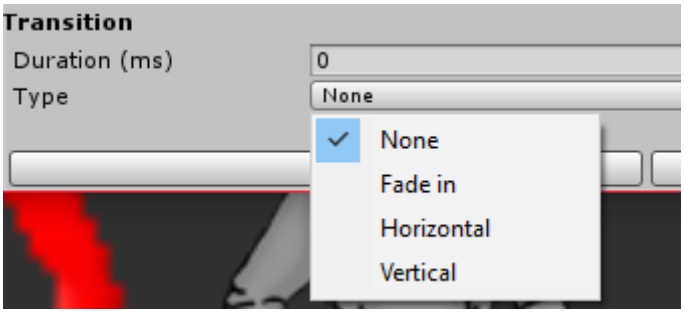


Figure 100. Properties of a transition in the animations editor.

At any point the "Preview" button can be clicked, and the animation will be played from start to end. OK and Cancel buttons are used to save or discard changes respectively.

3.9. Polygonal exits and active areas

Exits and active areas can be defined using complex polygons instead of rectangles. This facilitates using exits and active areas when the background image is complex. The usage is the same in both types of elements.

To activate this feature, select an exit or active area and switch to “Area” in the element inspector. Three new controls will appear behind the button (Figure 101). The first button “Move”, can be used to just move the vertex without performing any operation. The second, “Add” can be used to add a new vertex to the polygon. When adding points, a preview of the new connection is seen in blue lines (Figure 102). Finally, the third button, remove, can be used to remove a vertex.

In addition, when using the area mode, all the vertex handlers will appear in light blue, and will be circles instead of squares (Figure 102).

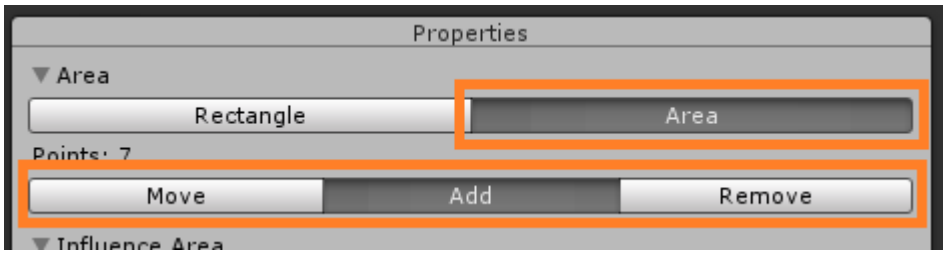


Figure 101. Area section in the element inspector. Area mode is selected. The tools are in the bottom: Move, Add (selected) and Remove.

Figure 102. Creation of polygonal exits in a scene.

Each vertex can also be edited by clicking and dragging them on the preview panel below.

3.10. Other features: error dialog

3.10.1. Error console

Although <e-Adventure> was a stable product and has been tested, its successor <u-Adventure> is a brand-new implementation of the engine and therefore it cannot yet be guaranteed that it is free of errors and bugs. Previous <e-Adventure> distributions used to have an error window that is not yet available in uAdventure. However, by opening the Unity console, many errors and log messages can be found to help developers find possible bugs. To open the Unity console, go to “Window > Console”. The three buttons in the right allow to filter the current messages and errors.

In case you find an error, it might be displayed here, so we suggest using captures of the console error as in Figure 103 to indicate the errors to the developers. If you want to share with us the errors you find you can either use the <u-Adventure> forums or go to our GitHub issues[12] section and open an issue in there attaching the picture of the console.

If you know how to reproduce a bug, we suggest you to first press the “clear” button before starting to avoid messing with older or non-relevant messages.

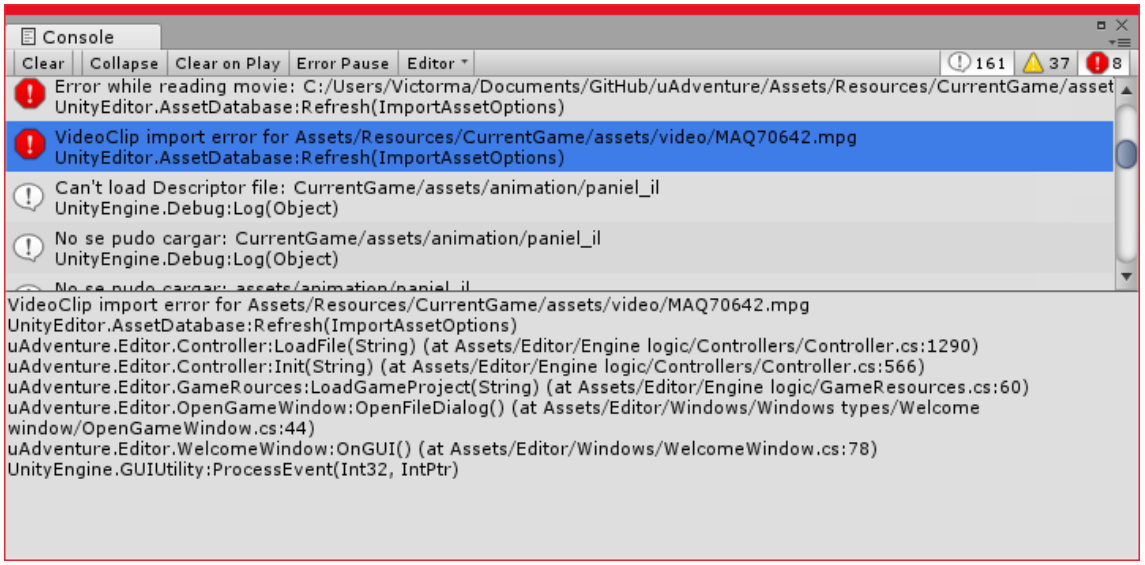


Figure 103. Unity console with an error selected.

In the future, the error dialog from <e-Adventure> might come back. For now, this console system is the easiest way to find errors. Thanks for your cooperation.

[12] https://github.com/e-ucm/uAdventure/issues

4. Geolocation features

This section describes the <u-Adventure> geolocation features. The geolocation comes up as a <u-Adventure> extension, and hence it might not be available in your <u-Adventure> version. Check your package and download the right version if you cannot find the Map Scenes or Geo Elements option in your main <u-Adventure> window.

4.1. Introducing the location-based game model

This section aims to be an introduction of the geolocation model features. <u-Adventure> is the new generation of <e-Adventure> and hence it takes advantage of the smartphone features such as camera and GPS location. Location-based games are sometimes referred to as Augmented Reality, but they include a broader set of games that does not fit in that category.

A location-based game uses geolocation as one of its main features. To be easily understandable for the player, this location is displayed on a map that is known as the augmented map as it includes both real-world features and game-based features. This differs to the standard augmented reality as augmented reality uses the camera to augment the image with game-based features instead.

In <u-Adventure> we have named these augmented maps as “Map Scenes” as they represent the mix-up of the concept of a <u-Adventure> scene and an augmented map. These scenes display the player in the center of the map, location-based features such as points of interest and regions that we have named “Geo Elements”, and <u-Adventure> elements such as characters and objects.

4.2. Map Scenes

As previously stated, Map Scenes are maps that include both real and game elements (Figure 104). These maps are also known as augmented maps and suppose a step in-between augmented reality and adventure games. The map scenes include an appearance configuration, a gameplay area configuration, a references section and a documentation useful for analytics and edition.

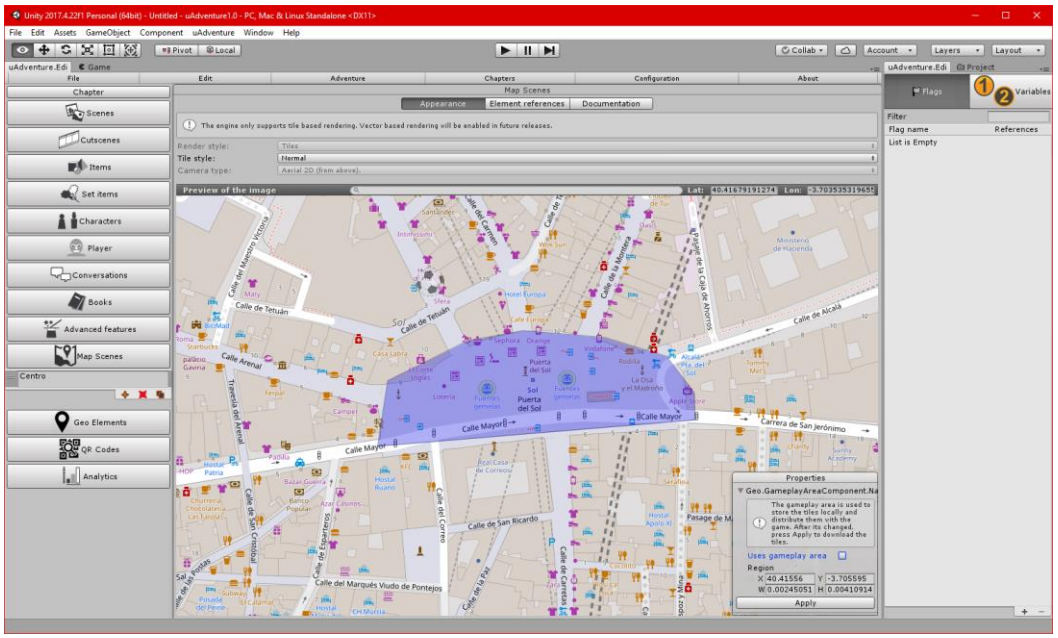


Figure 104. Map Scene in the appearance section, with a region in the center.

Map Scenes are created by pressing the "+" button, same as any other <u-Adventure> element.

4.2.1. Map Editor

In this section, before explaining map scene aspects, we will explain how to control the map editor. The map editor is shown in any map related feature such as Map Scenes and Geo Elements. This map editor includes a top bar in grey with a search field, a pair of fields with the current latitude and longitude and a tile-based map in the bottom. The tiles from the map are downloaded on-demand depending on the style selected (see 4.2.2.2) and cached[13] in the computer for one week till they are downloaded again.

When a map scene is created, a part of a world map is displayed. There are two options to determine the gameplay area where you are going to place your elements. First option is using the map controls. The map implements commonly-used map controls such as dragging and zoom. The drag control can move the map center around by clicking and, while pressed, moving the mouse around the map. The zoom control can change the distance from the map and can be changed by using the mouse wheel.

If you want to move faster in the map, the place searcher on top on the map (Figure 105) can help to easily find a specific location. After typing a location, a drop-down list will appear with some suggestions related to your query. By clicking in any of them, the map will go straight to that point.

Figure 105. Place searcher showing some results.

Another way of moving the map is by using the latitude and longitude fields. These fields can help you in a very technical way, if you know the exact coordinates from a smartphone or a GPS device.

4.2.2. Appearance

In the Map Scenes appearance section, we can find different options to configure our map. These options include the render style, the camera style and the tile style. In contrast to the normal scenes, we cannot choose the background itself, but we use the tile style to customize the appearance we want in our scene to represent the real world.

4.2.2.1 Render Style

The render style attribute defines how the scene is going to be rendered. For now, only the "Tiles" based rendering is available and it means that image tiles will be used as pieces of our map. This image tiles can be swapped between one style and another in the Tile Style attribute.

Future releases will include Vectorial based rendering and offer more options for customization and tridimensional rendering, such as choosing which element types are rendered and which colors

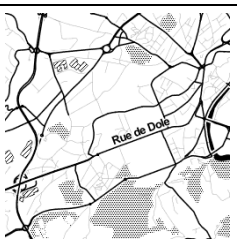
[13] In case you want to manage the tile cache you can check it in the temporal files folder of the project at "AppData/LocalLow/<Author>/<Game Name>/".

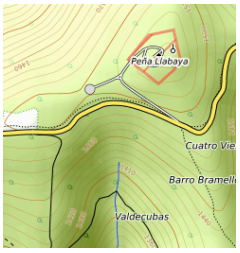
and materials will be used for each. These vectorial maps are based on vectorial tiles that are numeric based representation of elements such as regions or points of interest.

In addition, in the future a hybrid style will be implemented too, including both vectorial and image-based rendering at the same time.

4.2.2.2 Tile Style

The tile style defines which tile provider will be used and hence will change the appearance of the map and its tiles. The tiles available in <u-Adventure> come from two different providers in a total of five total representation and these might get extended in the future.

Tile Style	Description	Example
OpenStreetMap's	OpenStreetMap (OSM) is a collaborative project to create a free editable map of the world.	
OpenStreetMaps w/o Labels	A special style of OpenStreetMap tiles but without the street and feature labels that might interfere in the game.	
Carto Light	CARTO is an open, powerful, and intuitive platform for discovering and predicting the key insights underlying the location data in our world. CARTO offers their own rendered stylized OpenStreetMap tiles open for all to use for free.	
Carto Black	They currently offering a choice of two styles, one very light, and one very dark, with very little use of color. They are heavily stripped down simplified styles designed for overlaying data upon.	
Stamen Toner	Stamen is a world-leading provider of visualization and analytics design and strategic data communication for private and public sector clients across various industries. We offer two of their most interesting tiles:	

Stamen Watercolor	i) toner based on black and white with a toner aesthetic with low amount of features ii) watercolor, that is a rendering based on texture painting without any text or mark, nice for historical or cultural games	
Open Topo Map	OpenTopoMap is a project aiming at rendering topographic maps from OSM and SRTM data. They render topographical information such as paths, rivers, summits, heights, etc. It is a map interesting for hiking games.	

4.2.2.3 Camera Style

Camera style defines how the map is going to be displayed. For now, only the Aerial2D is available and it shows the map from a perpendicular view, pointing north, with no more controls (Figure 106). In future releases, more rendering like Perspective 2D and 3D will be added.

Figure 106. Map Scene in Aerial 2D view.

4.2.3. Gameplay area

The Gameplay Area defines a region which tiles will be downloaded and packed in the final game. This allows the game to be played without internet adding more flexibility for certain regions. To use a gameplay area, it has to be enabled in the sub editor window in the map (Figure 108). Once the area is enabled, a bright rectangle will show up of the size of 0.2 degrees in both latitude and longitude (Figure 107). This area size will be approximately 35 megabytes. Smaller regions will decrease the size whereas bigger will occupy more exponentially. Keep in mind that using the right amount of space is recommended when developing for Android as most smartphones easily run out of space.

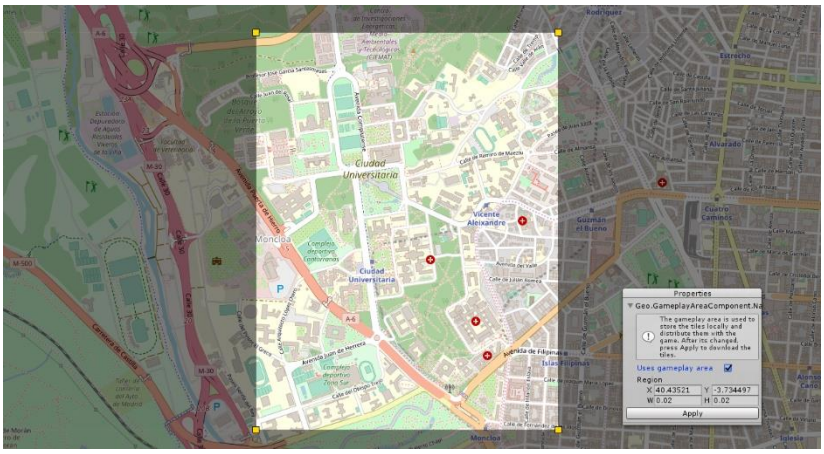


Figure 107. Gameplay area showing up in the middle of the map.

Finally, once the area is selected and "Apply" is clicked, the storing process will start by removing the unused tiles in case of modification of the area, and then downloading and storing the new tiles. A progress bar will indicate the state of the process. If the download process fails, the process will stop and an alert will show up.

Figure 108. Gameplay area context window.

4.2.4. References

To add gameplay functionality to the map scenes, other elements have to be added to the scene. These elements may include <u-Adventure> elements such as Items, Characters or Atrezzos and geo elements such as regions or points of interest. To add an element to the map, use the list on top of it. By pressing the "+" button a contextual menu will show up letting the user select the element category and then, the element itself.

Once the element is added, it will appear in the map at a scale proportional to the current zoom map. We recommend using scales from 17 to 19 (max scale) when placing elements in order to perceive the sizes that will be seen when playing the game. On the other hand, geo elements (see section 4.3) are natively placed as they represent map regions but, native <u-Adventure> elements must be positioned in the Map Scene any way. To choose between these positioning options, once the element is selected we will be able to switch between them in the inspector window on top of the map scene editor. We have determined three different positioning options.

To switch between the positioning options, once the element is added you have to select the element and you will be able to select between the available positioners in the inspector window on top of the scene (Figure 109).

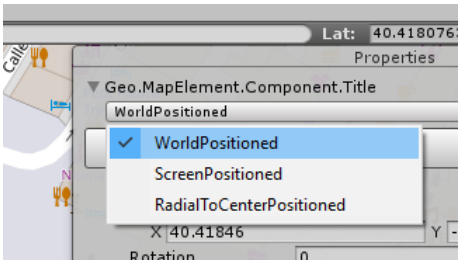


Figure 109. Dropdown menu showing the different positioners.

4.2.4.1 World positioning

The positioning based on geolocation consists in assigning the element to a specific set of coordinates (latitude and longitude). This element will be hooked to that location and as soon as the player moves in the map this element will remain in that location. If you want to easily go to the element location, press the " " button (Figure 110) either in the inspector or in the references list.

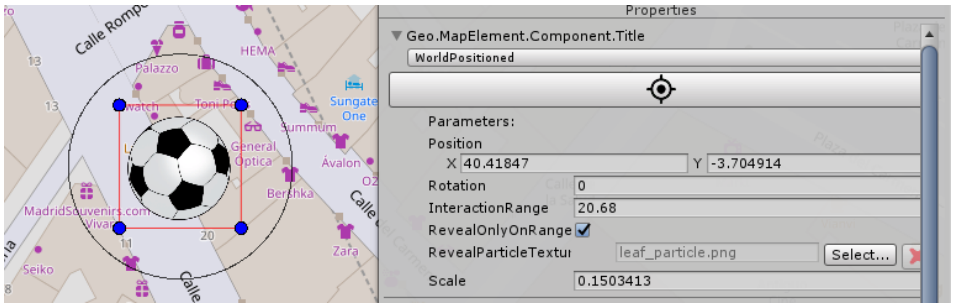


Figure 110. World positioned element selected in the map scene editor. In the right, the inspector shows the world positioned

Some options have been specially designed for this kind of positioning. The first option is the "Interaction region" that determines the distance, in meters, where the item is interactive or not. By default, the value is 25 meters and the measure will be displayed in the editor as a black circle around the element (Figure 110). This region is used to avoid users interact with certain elements unless they are close enough to them.

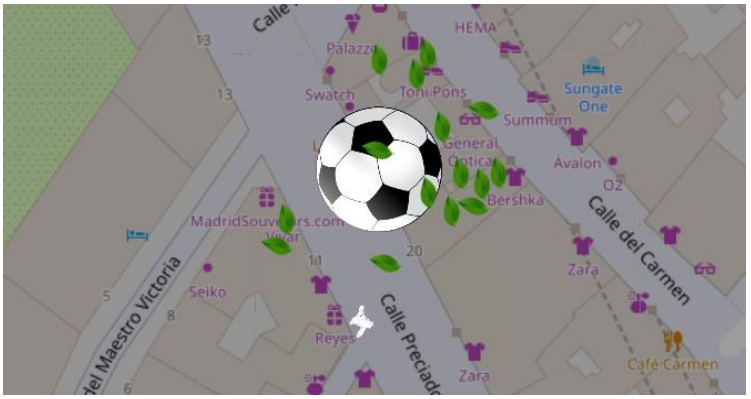


Figure 111. World positioned element being revealed.

Furthermore, if the "Reveal Only On Range" attribute is active, the element will only show up if

the player is inside of the Interaction region, remaining invisible otherwise. When the item shows up, an animation is displayed with some leaves popping up by default (Figure 111). However, the texture of the leaf particles can be changed in the "Reveal particle texture" attribute.

4.2.4.2 Screen based positioning

The screen-based positioning attaches game elements to a certain position of the screen. The elements will remain in the same position and scale independently of the player position in the map or the movements in the game camera. This positioning method is perfect for creating interfaces or companions, as they will always appear in the screen and the player could access them at any time.

4.2.4.3 Radial based positioning

The radial-based positioning attaches the game element to a world position relative to the player position, using a radius and an angle to calculate the final position. This means that the element will always appear at that relative position no matter the player moves but will be susceptible of camera movements or zoom changes as it will remain in the same world position when these changes happen.

This feature supposes an intermediate option between world and screen position, as the element will always be present in the screen but will be placed in the world. A good example of radial based positioning is a navigation arrow, which is shown in the world, but moves along with the player.

To use this positioning, an angle in degrees (0 to 360) and a radius in meters is used to determine the position instead of the classic pair of coordinates. In addition, the rotate around attribute can be used to rotate the element as the compass orientation changes.

4.2.5. Documentation

The documentation section of the Map Scenes is very similar to the one present in the Scenes. It has two fields for analytics purposes to define its xAPI Class and xAPI Type (explained in depth in the Learning Analytics section), a field for the name of the scene and a full description that is used for documenting the development process. None of this information will appear in the game and will be used either to describe information or to configure analytics.

4.3. Geo Elements

In map scenes it is possible to represent regions, points of interest or paths in a similar way as active areas or exits represent regions of a scene. These elements that are linked to the real world and isolate map features are called geo elements. In contrast to the previously mentioned areas, geo elements are created on top of the map, and hence, are shared among all the map scenes they are referenced in.

A geo element can have three different geometry types (point, polygon or string) representing points of interest, regions or paths. In addition, the geo-elements offer brand new interactions based on GPS and compass, linked to the real world and offering a more ubiquitous learning. Finally, geo elements can also be used in location-based effects explained in section 4.4.

The geo elements are created in the geo elements section by pressing the "+" button. However, to help you in the process of defining its area, once you have introduced a valid Id, the next window will ask you for an address or location. You can skip the selection by clicking on "create blank", but if you select a location, a rectangle surrounding the selected area will appear. In the next sections we will show you how to modify this area and add the different location-based actions to the element.

4.3.1. Geometry

The first section of a geo-element is its geometry. In this section we can manage its possible geometries as appearances for other <u-Adventure> elements. The element must have at least one

geometry to be visible (Figure 112). When more than geometries are available, these can be selected by using priority conditions in runtime, the same way as appearances work.

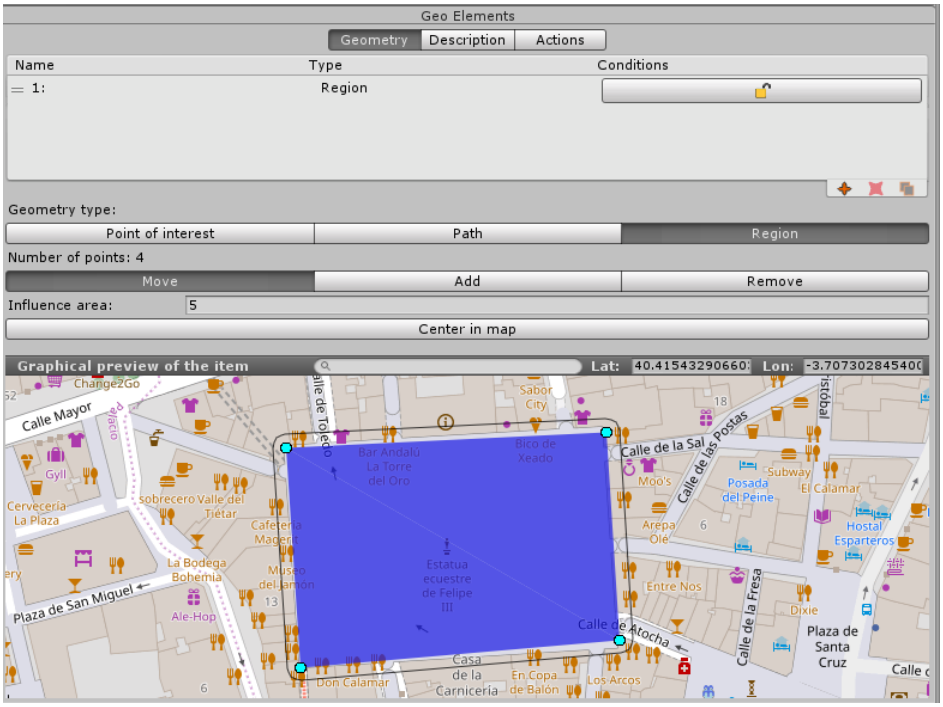


Figure 112. Geometry section showing a region.

There are three types of geometries (Figure 113):

- A point of interest is a geometry consisting of a single point and when a new point is added the new point just replaces the original.
- A path is a string defined by several points that are connected in order and left open from beginning to end. New points added to a path are always added at the end.
- A region (selected by default) is defined as a closed polygon, same as active areas, influence areas or exits. Regions must have at least three points, but there is no point limit for them. When adding a new point, it is added creating a new joint in the closest edge.

Note that if no influence area is set in points and paths, they will not be interactive as the chances for the player to just match the element location are minimum.

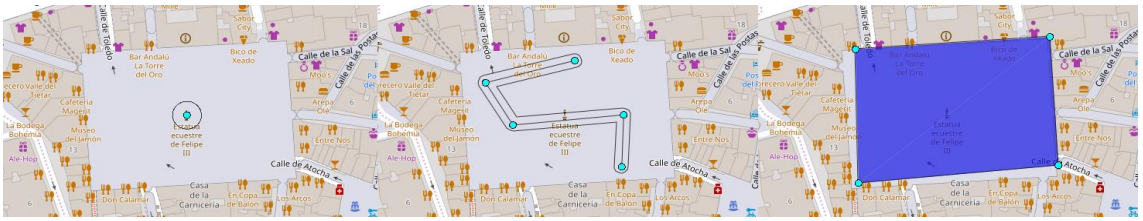


Figure 113. Different geometry types: point of interest (left), path (middle), region (right).

Managing the points is as simple as in active areas or exits. The geometry section offers a tool selector with three options: move, add and remove. By switching between them, you will see the points of the area changing its shape and color (Figure 114). This is just an editor help but will not

be visible in runtime. When moving, the points will be a round light blue point. When adding, small squares will identify the point and the mouse will show a small "+" button. Finally, when removing, the points will turn red and the point that will be removed will turn yellow on hover.

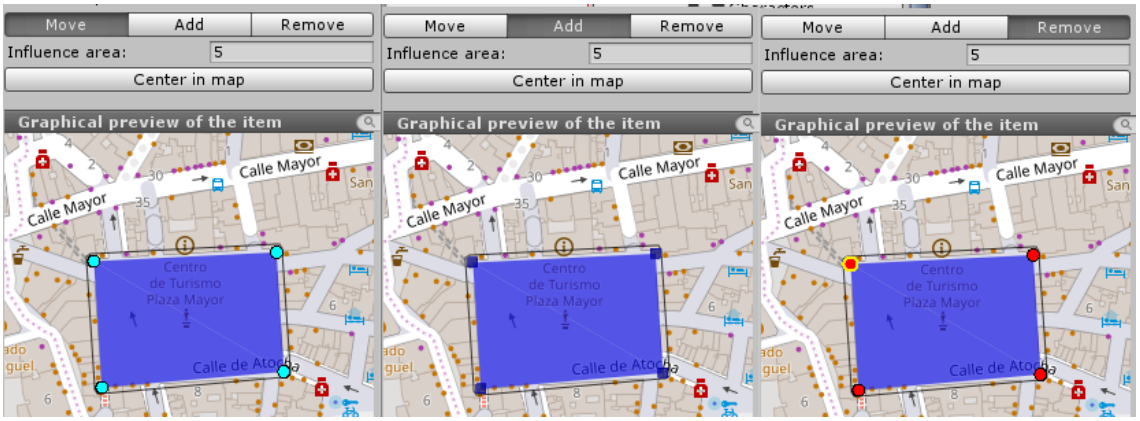


Figure 114. Geometry tools and point highlight.

The influence area of a geo-element is essential as it defines the minimum range where we consider the player is in contact with the geo element. It is defined by a measure in meters and represented as a black line outside the shape.

4.3.2. Description

The geo elements have a description setting to define its name and description and add documentation if needed. The geo-element can have multiple descriptions, and these can be controlled using conditions. Finally, if the geo-element contains a name, a tooltip will be shown in-game, on top of the geo element (Figure 115).

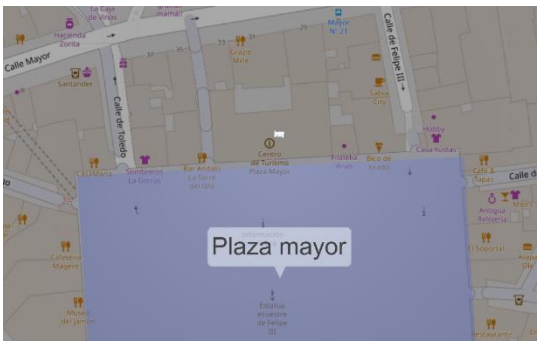


Figure 115. Geo-element tooltip.

4.3.3. Interaction with Geo Elements: Geo Actions

In the actions section of the geo-elements we can define special actions for location-based games. These actions depend on the player location and orientation by using GPS and compass and are four: enter, exit, look at and inspect. Other <u-Adventure> actions are not available for geo elements. To add a new action, you have to press the "+" button on the right-bottom part of the list and select the type of action you want to add. The next subsections will explain how each action works and how to properly configure their parameters.

4.3.3.1 Enter

The enter action is performed whenever the player is inside of the influence of the current geometry. For points of interest and paths only the influence is used, while for regions, being inside of the region will also trigger the effect.

There is a special parameter for the enter action named “From outside”. This parameter will make the action trigger only if the player was previously outside of the region. For instance, if the player opens the game and is inside of the region and this parameter is turned on, the action will not be triggered. This parameter can help you avoid re-entering areas on scene changes.

4.3.3.2 Exit

The exit action is performed whenever the player is outside of the influence of the current geometry. It works just the opposite way of the enter action (see 4.3.3.1). For the exit action there is also a special parameter named “From inside” that is especially relevant in this case as most of the time players will start the game outside of a region and you will want the action to be performed only after the player leaves the influence.

4.3.3.3 Look to

The look to action is performed when the player direction determined by the compass is heading to a certain point or direction. In case that the parameter “to center” is active, the action will trigger when the player is looking to the center of the geometry. However, if the “to center” parameter is off, a direction selector will appear, showing a white circle with a red line indicating the direction the player should look to (Figure 116). This direction represents the heading of a compass, so when it is heading to the player will have to head north; when is heading right, east; when is heading down, south; and finally when heading left, west.

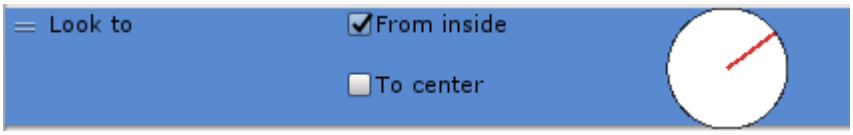


Figure 116. Direction control of the look to action.

Finally, the “from inside” parameter defines if the action is only triggered if the player is inside of the influence of the element. Hence, you will be able to control if the player heads to a specific location if you define the region to look from and the direction to look towards.

4.3.3.4 Inspect

The inspect action is performed when the player clicks (or touches) the region, and can be used to show information, descriptions, etc.

4.4. Geo-related effects

The location-based expansion also includes effects to use in a map-scene. However, these effects are passive, as they run things that react on location changes. There are two effects, a special trigger scene linked to a region and the navigation related effects. These effects can help the player through the map experience and create a hybrid between map mechanics and adventure mechanics.

4.4.1. Triggering scenes linked to regions

To create a hybrid experience between a map scene and a point-and-click adventure scene it is possible to trigger a scene linked to a region. This scene will remain open while the player is still in the area but will close if the player leaves the influence area.

To create this effect, at least a geo-element and a scene are needed. Then, when selecting the effect type choose “trigger zoned scene effect”. The editor will look the same as a trigger scene effect, but it will have the name of the zone at the end. All the other aspects of the trigger scene effect that are inherited from it work as detailed in section 3.1.10.

4.4.2. Navigation

The navigation effects control the in-game map scene navigation tool (Figure 117). This navigation helps the player moving through the map towards the selected elements. The goal of the navigation is not to find the best way to reach n specific location but to show the direction where the next game-relevant feature is.

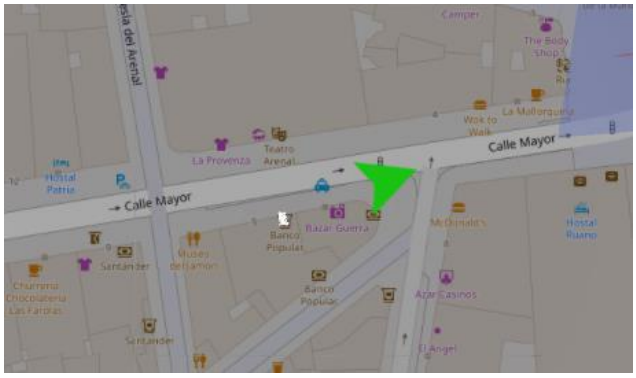


Figure 117. Navigation system tool.

The navigation system is controlled by two effects: “navigate” (Figure 118, left) to determine the steps in the navigation and “navigation control” (Figure 118, right), to progress the current step we are in. The navigate effect is launched once and sets up the navigation. This effect contains a navigation type and a set of steps. There are two navigation types: ordered or closeness. The ordered navigation will go to all the elements in the list in order. On the other hand, in the closeness navigation, elements will be sorted by proximity to the player and the arrow will head to the closest element.

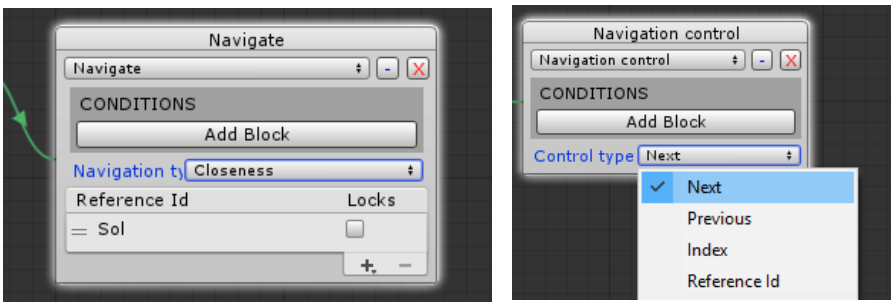


Figure 118. Left, navigate effect by closeness; right, navigation control and options.

The steps in the navigation are followed differently depending on the element and configuration, but in general the steps will progress automatically when reached unless they are marked as “lock”. Regarding the navigation towards the current element, the arrow will always head to the center of the element in any element except for paths, which can be used to define routes. The navigator will progress as soon as the influence is trespassed. Whenever a path is used in a navigation, the navigator will head to each of its points individually and guarantee the player reaches every point before the path is finished. This feature is useful in locations where we want to specify certain routes to follow.

When elements are marked as “locks” the navigator will not progress unless a “navigation control” effect is launched. This can be used to wait until the player completes certain tasks in the location and then, point to the next important location.

4.5. Debugging location in map-scenes

To play location-based games, it is necessary to have GPS connection. However, at edition time this is not needed as it is possible to simulate the players’ movements. Once a map scene is opened in the play mode, a small window will appear on the top-left of the screen showing the current player location (Figure 119). To change the player location, you have to click and drag the blue dot in the middle of the debug window. Once you do it, you will see the players’ avatar moving in the map towards the new location. This drag will activate the debug location mode and from now on, location will be simulated based on the position inside of this map.

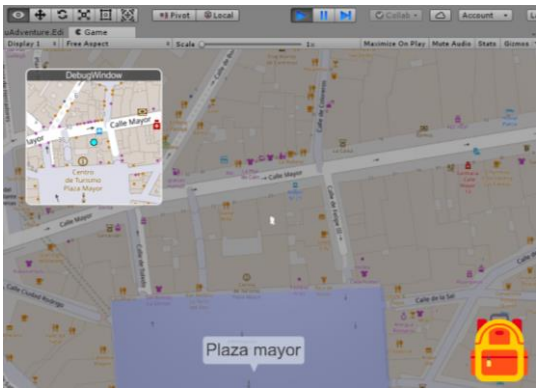


Figure 119. Map scene in the editor showing the simulated location window.

Thanks to the debug location window now you can simulate the player movements, including enter regions or exits regions when a “zoned scene” is launched.

5. QR Codes

In this chapter, we describe the QR Code <u-Adventure> features. The QR Codes are a type of barcode that can be used to encode information. Using a camera (e.g. web, phone) it is possible to capture the QR codes and decode its contents. These contents can be linked to <u-Adventure> and hence, they could be used to start effects, scenes, conversations, etc. In addition, since the player must physically use the camera to decode the QR, we are encouraging the player to investigate the real world and find QR codes to progress through the game. This offers opportunities for gymkhanas or tours that require the player finding a specific feature where the QR is hidden.

In <u-Adventure>, QR codes are created in the "QR Codes" section with an id (that is encoded in the QR) and a set of contents including text that will be displayed when the QR is scanned, effects that will be played and conditions to guarantee the QR code can only be captured in the right moment. Once all the QRs are created, the in-game QR code scanner can be launched through effects in the specific moment we want the player to find a specific QR code. In the next sections we will explain in depth the features of the QR code editor and scanner.

5.1. QR Code editor

The QR codes are managed in the QR Code editor (Figure 120). This section can be found in the model panel at the left of the editor, as any other <u-Adventure> structural element. To create a QR at least an Id is needed. This Id is then encoded into a QR that will give access to all its contents such as text, conditions or effects. Note that if the Id is changed the QR code representation will also change and thus the previous printed versions will no longer provide access to the contents.

The contents of a QR code include three parts: the text content, the conditions and the effects. Whenever a QR code is found, its conditions are checked. Then, the text content, which can be modified on the top of the editor, will be shown and the effects will be executed. The documentation field in the QR codes is used for developer information purposes.

Figure 120. QR Code editor.

Additionally, the QR code editor has two buttons to save and print the QR. When pressing the

"save" button, a prompt will be launched to create the file. This will generate a ".png" image with the QR code. On the other hand, the "print" button will show a printer prompt to select the printer and configurations and finally print the QR code.

5.2. QR Code In-game scanner

To access the camera in the middle of the game and be able to scan the QR code, <u-Adventure> includes a QR scanner that will control que QR code execution. This in-game scanner guarantees that the gameplay will not be interrupted by switching between apps in order to scan the QRs.

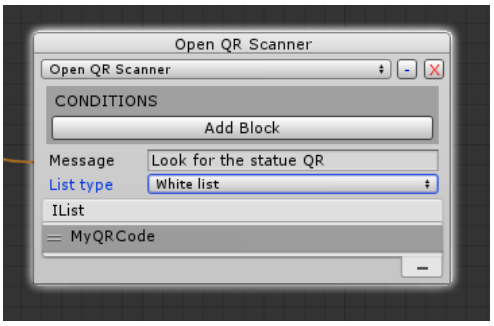


Figure 121. Open QR Scanner effect in white list mode.

The QR scanner is launched through the "Open QR Scanner" effect (Figure 121). This effect includes a message to give a hint to the player on the top of the screen and contains a list to filter the QRs available to that moment. There are two types of filtering, black or white list. By default, the list is in black list mode, so the QR codes in the list will be excluded from the scan. This means that by leaving the list empty the scanner will process all available QRs that match their conditions. On the other hand, by using the white list only the QRs in the list will be processed.

Figure 122. In-Game QR Code scanner.

In the game, the QR code scanner will appear on full screen mode using the back camera by

default (Figure 122).

6. Learning Analytics

Learning Analytics offer a new way of understanding how players use your game, learn through it and progress in it. With these Learning Analytics, both analysis of in-game actions and assessment of players can be performed in a meaningful way that is useful for teachers, developers and players. Learning Analytics suppose an evolution of previous assessment systems as SCORM. Instead of having to set up multiple variables to obtain measures of the interactions in the game, Learning Analytics rely on big data trends, were most interactions are traced and then queried and analyzed to extract value from them.

The language used in uAdventure to collect the Learning Analytics data is called Experience API (xAPI). This language is specialized in being an abstraction of real interactions where there is an actor, an object and a verb in each trace. In uAdventure, where there is only one person playing the game, there is only one actor known as *player*. In contrast, there are multiple verbs and object types in the xAPI standard, and uAdventure mainly uses the verbs and objects from the xAPI Profile for Serious Games (xAPI-SG) and the xAPI Profile for Location-Based applications (xAPI-LB). The xAPI traces are written in JSON, following their specific scheme, and uAdventure offers CSV conversion support for easier analysis. Examples of the different traces and formats can be found in section 6.8.

There are four categories of verbs and objects used in uAdventure: *accessibles*, to where the player can access (e.g. stages, scenes, levels); *gameobjects*, to which the player can perform actions; *alternatives*, to choose from; and *completables*, to progress in and complete. Additionally, for location-based games there is a specific profile to trace player movements, location-based actions and navigations. Further description of each object type, verb and additional parameters of each trace can be found in the later subsections.

Since Learning Analytics trace *player* interactions, we can provide traces out-of-the-box with very little configuration that represent most of the players' interactions and movements through the game. In fact, *scene accesses*, *element interactions*, *variable/flags changes* and *conversation choices* are available in <u-Adventure> to be traced with little to no configuration at all. These traces allow to get information after players finish the game, but also to further analyze how they played and what differences exist between players.

In some other cases, we may want to give meaning to specific interactions. This meaning can be translated into insights about the progress and the score in the game and the different tasks that must be accomplished in the game. For instance, we can infer from entering a specific scene that the player has completed a percentage of the game. This can be configured through completables that will be explained later in section 6.5.

The next sections explain the configuration required to collect traces of specific players' actions in the game and how to store and connect the learning analytics with external servers.

6.1. Scenes

Scenes are by default identified in the *accessible* class and hence they can be one of the following object types: *area*, *screen*, *zone*, *cutscene* and *accessible* (in general). This metadata can be configured in the "documentation" tab of the scene (Figure 123). When a player enters the scene, an *accessible accessed* trace is generated, with the scene Id and indicating the specific the scene type.

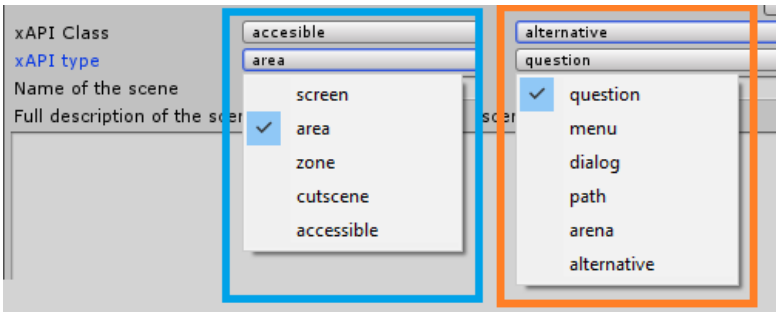


Figure 123. Scenes classes and types selector at the documentation tab.

It is possible to change scenes into *alternatives* and use their exits as choices in the game. Once changed in the "documentation" tab, now the type of the element can be selected from: *question, menu, dialog, path, arena* and *alternative* (in general). Then, once an exit is selected an *alternative selected* trace is generated using the scene Id and the scene type. In addition, if the selected exit id is appended as the option Id and the exit conditions match, the choice will be marked as successful (correct).

6.1.1. Cutscenes

The cutscenes have a special subset of traces. By default, a cutscene is configured as type *cutscene* and as with any *accessible*, a *cutscene accessed* will be generated. However, in addition to the normal scene behavior, if the cutscene type is left as *cutscene* it will launch a *cutscene skipped* trace if the cutscene is skipped, for instance, in the case of videos.

6.2. Game objects

When an element in the game is interacted a *gameobject interacted* trace is generated. These traces include the element id and the type of the element, that can be either *NPC* if interacted with a character, or *item* when interacting with an object. In contrast to previous traces, *gameobject* traces cannot be switched to other types or classes.

6.3. Conversations

In conversations, it is very common that the player must choose between different conversation options, to answer or continue the conversation. These options can be traced by setting up the option node parameters. To add some tracking information to an options node (e.g. if the node is a quiz question made to the student), we will have to identify the question with a unique "Question ID" and mark in the option checkbox which answers (choices) are the correct ones (Figure 124).

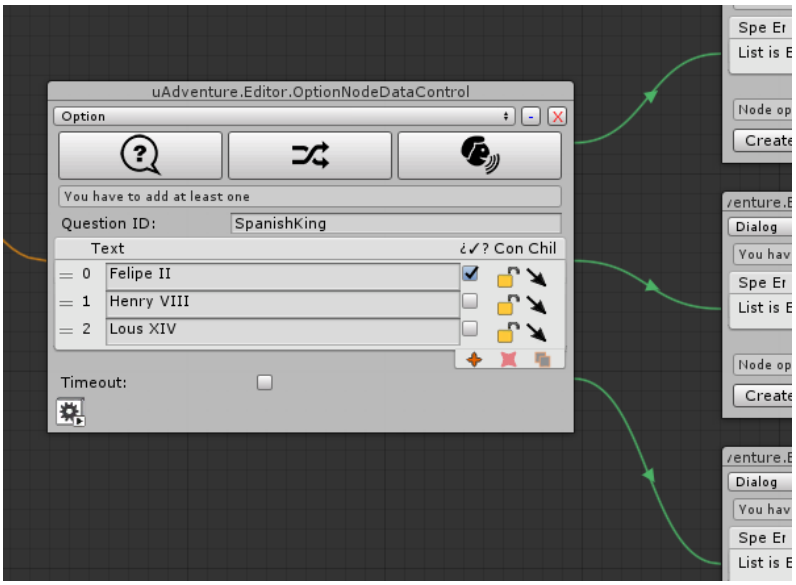


Figure 124. Question configuration about Spanish kings. The first answer is marked as correct.

Once the player selects an answer, an *alternative selected* trace is generated with the question Id, the text of the answer and the success extension if the option selected was marked as correct.

6.4. Variables and flags

The variables and flags changes are also traced. However, they do not conform a trace by themselves, but they are appended to the parent interaction trace that will generally be an *interacted* trace. For instance, if using an item changes some variable, the variable change will be appended to the current interaction trace and therefore the trace will be generated after the execution of the effect's finishes. This way, we will be able to generate richer and more valuable queries. When using the xAPI format, the variable changes will be appended to the end of the trace in the result extension.

There are some special cases where this feature can cause unexpected side effects in the trace reception or the location of the changed variables:

- Triggering a conversation can lead to *alternative* traces, in which cases, the variable changes after the alternative selected will be appended to the selected *alternative* trace.
- Triggering a scene will generate an *accessed* trace that will be sent right after. Then this trace will be received before the *interacted* as there may be effects after that could change variables.
- Changing a variable can lead to *completable progress* (or *started* or *completed)* in which case, this trace will be generated and sent before the *interacted* trace.

Some other side effects may happen not noted here, so please take this into consideration when planning your traces.

6.5. Completables

Completables offer a high-level progress measurement for traces. All the completables have the same state flow, that is based on milestones. In particular, completables start when a milestone is hit, they progress as different milestones are hit, and they end whenever another milestone is hit. Milestone configuration is explained in section 6.5.1. In addition, a *completable* that has been completed will have a *score* value (see section 6.5.3), that can be later analyzed or even used for assessment. Completables can be of one of the following types: *game*, *session*, *level*, *quest*, *stage*,

combat, *storynode*, *race*, and *completable* (in general). The different completables in the game are configured in the completables editor (Figure 125) inside of the "Analytics" section in the left structure panel.

In uAdventure there is always at least one *completable* of the type *game* that starts in the beginning of the game and ends when the game is finished. In this case, this *completable* has pre-configured *milestones*: it starts when the first scene is accessed and ends when the last chapter ends (this can be configured in the "Ends When" field). The progress of this *game completable* is calculated based on the amount of *completables* that are completed within the gameplay.

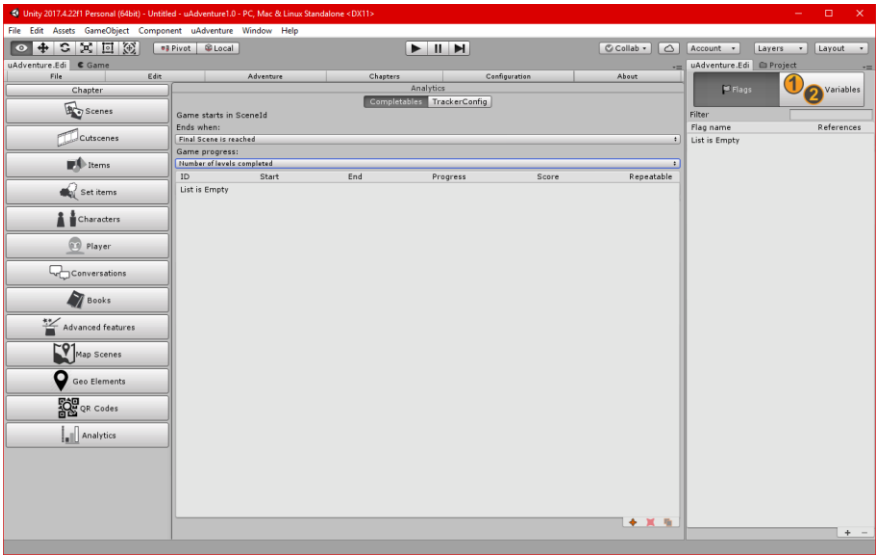


Figure 125. Completables editor.

Other specific *completables* can be added by using the "+" button in the bottom part of the *completables* list. These *completables* allow to track different parts of the game. For instance, in the "Fire protocol" game described in the examples, there could be one *completable* for each of the three different tasks of the game: 1) check the fire, 2) check the alarm, and 3) check and evacuate the people in the office. By adding these specific *completables* we can understand the behavior of the players, its performance during the gameplay, and their degree of completion of the different tasks.

The following subsections explore the different configurations of *milestones*, *progress* and *scores* of the specific completables.

6.5.1. Milestones

Completables can be divided into multiple steps or milestones. These milestones are reached from the initial *completable initialized* trace, generated at the start, to the final *completable completed* trace generated at the end. The conditions that lead to reach every milestone can be configured in the editor (Figure 126). Milestones can then be used to track the progress made in the *completable*.

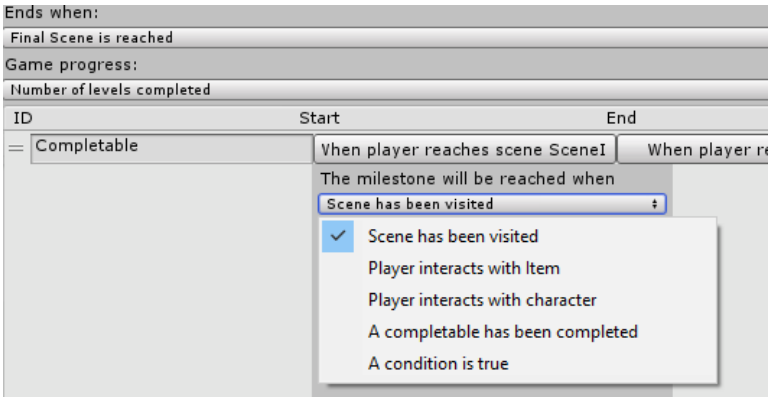


Figure 126. Milestone types. Each type has a specific content editor.

There are five types of milestones that simplify the usage of completables:

- Scene has been visited: When the player accesses a specific scene, this *milestone* will be satisfied. Note that if the scene has been visited prior to the *completable* being started, this milestone will not be satisfied until the player access the scene again.
- Player interacts with item: By performing any action with an item this *milestone* will be satisfied. For the *milestone* to be satisfied only when a specific action is performed on the item, the "a condition is true" option should be used.
- Player interacts with character: This action behaves like the previous "Player interacts with item", but for characters.
- A completable has been completed: This milestone is triggered as another completable reaches its end milestone, and hence, it becomes completed. This type of milestone can simplify the task of chaining different *completables*.
- A condition is true: This type of milestone is satisfied when certain conditions are satisfied. This is the most flexible type of milestone and should only be used when the previous milestone types are not enough to satisfy the needs of the completable.

6.5.2. Progress

The progress in the *completables* allows introducing another step in between the *start* and *end* traces. The *progress* works by linking different gameplay *milestones* with a competition amount. For instance, a *completable* can have four *milestones* that will add 25% to the competition *progress* each time they are satisfied. A *completable progressed* trace is generated with the progress as extension, when each of the *progress milestones* are satisfied. The *progress* can be configured by using the defined *milestones* (section 6.5.1), and can be configured to be calculated by: either the percentage number of satisfied *milestones* out of the total, or the maximum progress value out of the satisfied *milestones* (Figure 127).

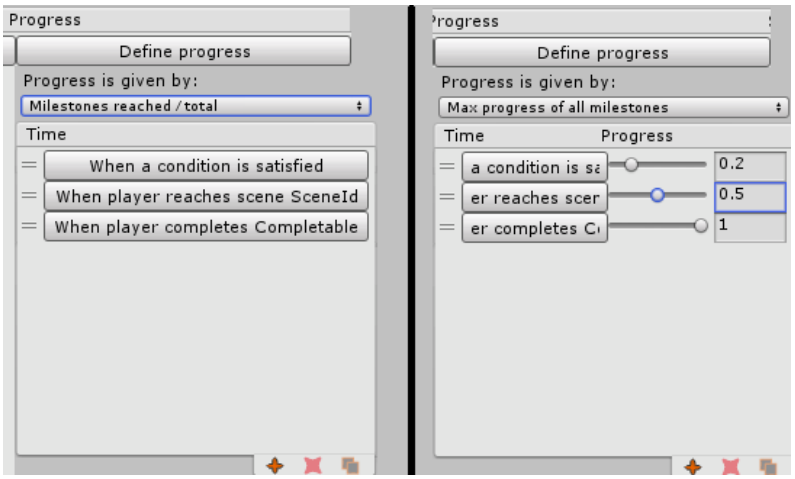


Figure 127. Completable progress types with three milestones. The left will be calculated depending on the amount completed whereas the right will indicate the greatest of the values.

6.5.3. Score

When a *completable* is *completed*, it can have a *score*. The calculation of the *score* obtained in a *completable* can be configured using the score editor (Figure 128). The single score calculation can be either a single value including *variables* (see 3.1.2). or previous *completables scores*.

The score will be traced as a result extension in the *completable completed* trace.

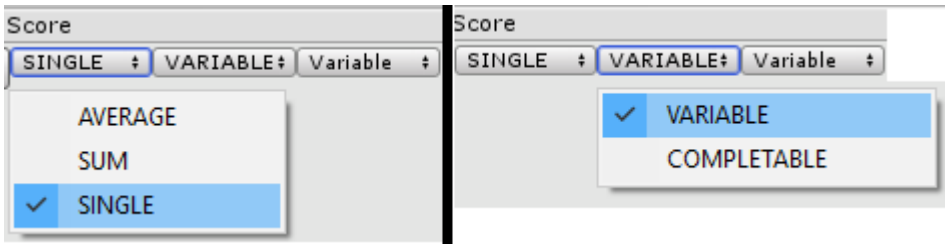


Figure 128. Score calculation types (left). Single score types (right).

Complex scores calculations that need to use several variables can be configured in the sub-scores editor (Figure 129) when any of the complex score type is selected. In general, the complex scores take their sub-scores values and generate a combined value by performing an operation. There are two types of operations, the *sum* and the *average*. The *sum* will add the different sub-scores as its *score* value whereas the *average* will sum them but later divide the result by the number of sub-scores. This system calculation is recursive, and hence, infinite sub-score layers can be stacked to create the *score* that suits the needs of the *completable* design.

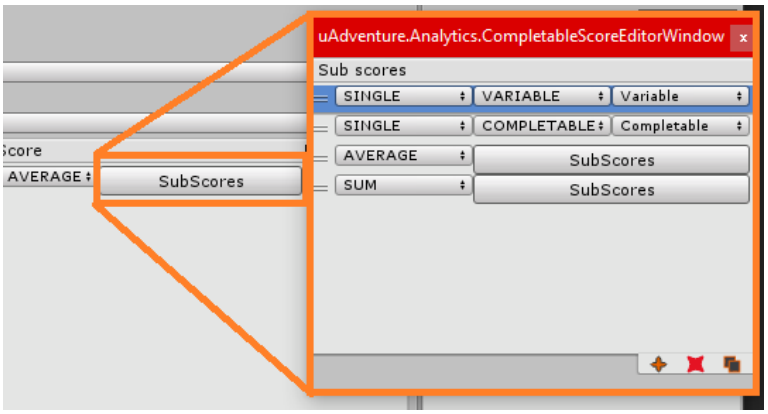


Figure 129. Sum or average types sub-score editor.

6.5.4.Repeatable

By default, *completables* will only be *started* and *completed* once during the gameplay. However, the *completables* can be configured as repeatable. When a *completable* is repeatable, it can be re-*started* after it has been *completed*. To repeat a completable, the player has to satisfy the start milestone again, and it will not be repeated again until it is completed.

By repeating a *completable* it will generate the same *traces*, but as the repetition happens after the first *completion*, the *timestamp* parameter of the *traces* will be higher than the previous runs.

6.6. Geolocation

There is a special set of traces regarding location-based games. These traces include positioning for the player, orientation, traces for the location-based interactions (geo actions) and navigation. In this case, the location-based profile for serious games is used and it includes the verbs *moved, entered, exited, looked* and *followed.* Using these verbs, the different traces are generated.

6.6.1. Player location changes

Trace with the verb *moved* are used whenever the player moves inside a map scene, using the map scene id as the target and the *area* type as object type. In addition, this trace includes as a result the position and, optionally, the orientation of the player if available. This trace, in contrast to others, is generated every few seconds to be able to follow the player movements around the map and the time spent in the different locations.

6.6.2. Interactions with geo elements

The analytics for location-based includes traces for the cases where the player performs location-based actions with geo-elements. In section 4.3.3, we described four types of interactions, including three purely based on player's location or orientation. These actions are *enter*, *exit* and *look to*. These actions are being traced with the special verbs in the location-based profile.

First, the enter action will generate an *entered* trace, using the object type *region* or *point-of-interest* (if the geometry is a point) and the geo-element Id as identifier of the target. As result extension it will incorporate both location and orientation (if available). The latitude and longitude is expressed in geopoint format[14], and the orientation in degrees.

Second, the exit action will generate an *exited* trace using the object type *region* or *point-of-interest* and the geo-element Id. This trace will also incorporate the location and orientation as result

[14] https://www.elastic.co/guide/en/elasticsearch/reference/current/geo-point.html

extensions.

Lastly, the look to action will generate a *looked* trace using the same object types as both previous traces and in this case the orientation will always be in the trace.

6.6.3. Navigation

Once the navigation is enabled and after a navigation step is completed, a *followed* trace using the type *direction* as object is generated. Finally, the results extensions will incorporate an extension *guide* with the full set of steps the player is following.

6.7. Tracker configuration

Once the interactions that are to generate the xAPI traces are configured, for those traces to be saved and stored, the tracker needs to be configured.

First, the storage type can be chosen between local and net. If local backup is selected, traces will be store in a specific location depending on the operating system:

- **In Windows, the traces location will be:**

```
C:\Users\<username>\AppData\Local\Temp\<author>\<game-name>
```

- Alternatively, this location can be accessed as:

```
%TEMP%\<author>\<game-name>
```

- **In Mac, the traces location will be:**

```
~/Library/Caches/unity.<author>.<game-name>
```

- **In iOS, the location will be:**

```
/var/mobile/Applications/<package-name>/Library/Caches
```

- **In Android, the location will be:**

```
/mnt/sdcard/Android/data/<package-name>/cache
```

- **In Linux, the location will be: [TO-DO REVISAR]**

```
~/.cache/ or /var/tmp
```

The parameters <author>, <game-name> and <package-name> of the location can be configured in the build window of uAdventure as detailed in Section 7.1.2.

The traces format can be either CSV or xAPI. See Section 6.8 for examples of traces in both formats.

For net storage, the hostname and tracking code need to be configured.

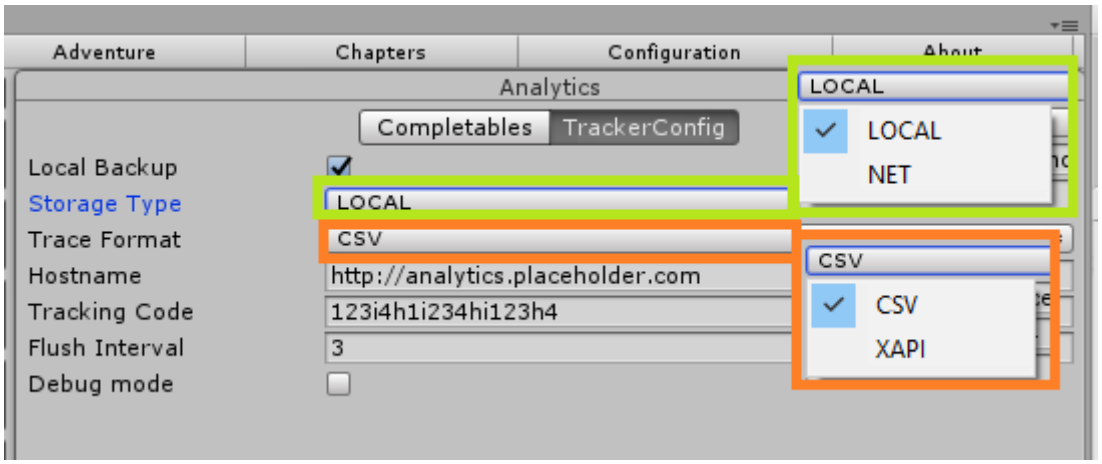


Figure 130. Tracker configuration.

6.8. Traces examples

As described in the previous section, the traces collected in uAdventure may be in format CSV or xAPI. In this section, we provide some examples of traces in both formats to clarify how the information is stored in such formats.

6.8.1. CSV

When traces are stored in CSV format, each trace corresponds to one row in the CSV file. The first column in each row (i.e. trace) will store the timestamp of the trace in Unix time[15], in milliseconds. The second column will contain the verb. The next columns will contain, in order, the object type, the object identifier, and all the extensions included, storing, for each extension, the key and the value, also separated by commas.

For example, consider the event of the player interacting with the "Phone" object in the game. When interacted, the player selected the "talk to" action and during the conversation that this action triggers, the number 112 is selected as an option that finally turns on the flag "Llamo112". The trace generated will include both the "iteracted" verb, the "item" object type, the ID "Phone" and then two extensions that include the action type and all the flags and variable changes. All the extensions are included after the element ID and are pairs of key-value. The corresponding CSV trace generated will be the following:

```
1591595043271,interacted,item,Phone,action_type,talk_to,Llamo112,1
```

6.8.2. xAPI

When traces are stored in xAPI format, the JSON file will store the xAPI traces one after the other, separating them with a comma. In this case, timestamps are stored using the ISO 8601 format[16].

The previous example trace will be stored in xAPI as:

[15] https://en.wikipedia.org/wiki/Unix_time

[16] https://en.wikipedia.org/wiki/ISO_8601

```
{
  "actor": {
    "name": "Player"
  },
  "verb": {
    "id": "http://adlnet.gov/expapi/verbs/interacted"
  },
  "object": {
    "id": "http://a2:3000/api/proxy/gleaner/games/<game-id>/<version-id>/Phone",
    "definition": {
      "type": "https://w3id.org/xapi/seriousgames/activity-types/item",
    },
  "result": {
    "extensions": {
      "action_type": “talk_to”,
      "Llamo112": "1"
    }
  },
  "timestamp": "2018-06-11T08:34:08.191Z"
}
```

As a summary, Table 1 shows the game events traced in uAdventure, describing for each event, the cause that triggers it, the xAPI type and verbs used to track it, the target and the possible results stored (including response, success and other extensions).

Event	Cause	xAPI Type xAPI Verb	Target	Result R: response, S: success, Ext: extensions
NPC Interaction	Player opens NPC actions menu	Character Interacted	NPC name	Ext: Action name
Item Interaction	Player opens item actions menu	Item Interacted	Item name	Ext: Action name
Scene access	Player enters a scene	Accessible Accessed	Scene id	
Cutscene start	Player starts a cutscene	Cutscene Accessed	Cutscene id	
Cutscene skip	Player presses skip	Cutscene Skipped	Cutscene id	Ext: Percent watched
Player moves	The player changes his GPS coordinates by moving	Place Moved	MapScene id (*“World”)*	Ext: Position Lat Lon
Geo Element Interaction	Player enters, exits or looks a Geo Element	Place Entered, Exited, Looked, Interacted	GeoElement id	Ext: Position Lat Lon Direction in degrees
Element discovery	Player finds an element in the map scene	-	-	Ext: geo_element_<id>
Navigation	Player follows navigation	Followed Direction	Next target id	Ext: guide of steps
Exit selection in alternative type scenes	Player selects an exit in current scene, for menus or visual choices	(Alternative, Question Menu or Path) Selected	Exit Id	R: Arriving scene S: Based on exit conditions
Dialog choice	Player selects one dialog option	Alternative Selected	Question Id	R: Response S: Correctness
Completable start	Player reaches a milestone	Completable Started	Completable Id	
Completable	Player reaches one of the	Completable	Completable	Ext: Milestone progress value

progress	milestones	Progressed	Id	
Completable finish	Player reaches a milestone or completes all the steps	Completable Completed	Completable Id	R: Score from variable S: Based on conditions Ext: Time
Game start	Player visits title	Game Started	Game name	
Game progress	Accomplishment of any of the levels	Game Progressed	Game name	Ext: Progress as percent of levels (tasks) completed
Game end	Milestone or all levels completed	Game Completed	Game name	R: Avg. score of all levels S: Based on conditions Ext: Time

Table 1. Summary of game events traced in uAdventure

7. Menu options

In this section the menus on the window bar will be explained. These include common features such as management of games and projects and other characteristics that allow further customization of the games.

7.1. File menu

This menu supports basic options to manage files, create new game projects, export games, exit, etc. Most of these features have been described in section 2.

7.1.1. Save

To save the uAdventure game, under the file menu select the "Save" option. Also, whenever the game is played, if modified, a window will pop-up asking you if you want to save the game (Figure 131). It is important to save the game each time as uAdventure model is not saved until a save is explicitly performed. Users may be aware that by using the Unity's File->Save or File->Save project options the uAdventure model WILL NOT BE SAVED.

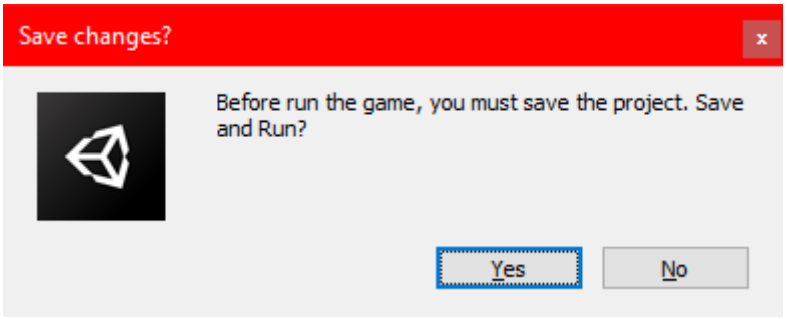


Figure 131. uAdventure save prompt

7.1.2. Build window: startup options

There is a special case in the build window to give the control of how the game will be run in the standalone platform. The startup dialog is a Unity feature that lets the player select the resolution, the graphics quality, the windowed mode and even the monitor that is going to be used (Figure 133).

This panel is, however optional, and its usage can be configured depending on three options in the Build window (Figure 132):

- Hidden by default: is the default option. The panel will not be displayed unless the "Shift" button is pressed during the startup.
- Enabled: The panel will be shown normally.
- Disabled: The panel will not be shown and the user will not be able to open it.

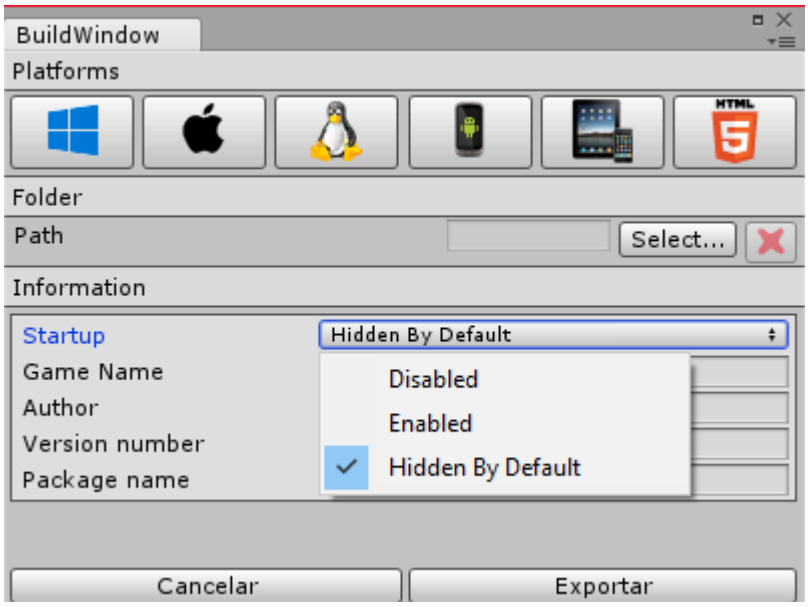


Figure 132. Build window, startup configuration.

In this window, you can also define the game name, the author, the version number and the package name. Both the game name and the author are used to define the location where the traces will be stored, if analytics in local mode are configured (as detailed in Section 6.7). The package name expects a name like “com.author.game”. No special characters are allowed.

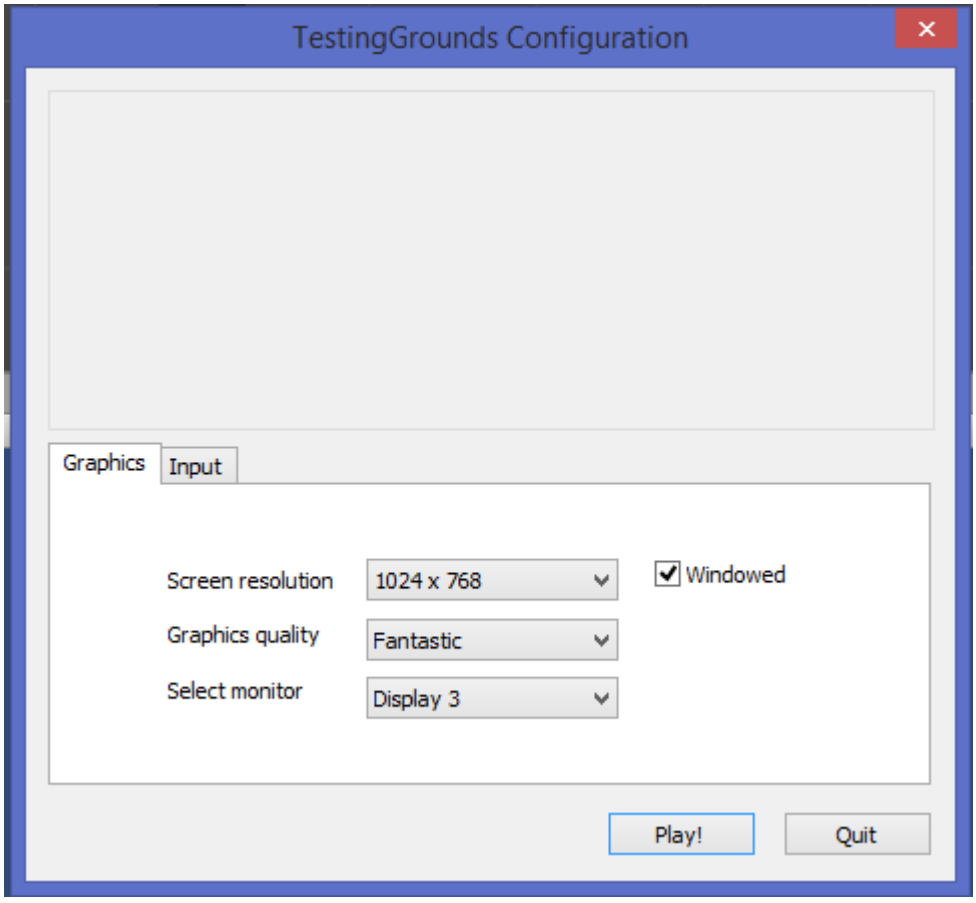


Figure 133. Unity resolution and quality window.

7.2. Adventure menu

This menu allows editing different aspects and provides general information of the game and allows customizing some of the elements of the user interface in the games: cursors, action buttons and the inventory. Also, the save and load behavior can me modified.

7.2.1. Adventure info and main settings

In the settings tab (Figure 134), the adventure title and description can be modified. Also, the game mode can be checked. In contrast to e-Adventure, in uAdventure some options are not yet implemented. These options are:

- Changing the game mode from 1st to 3rd person and vice versa.
- Default left click action is always open the actions menu, unless first action mode is specified in the element. This has been implemented like this to facilitate touchscreen input over mouse.
- Perspective is always regular.
- All elements are targets for drag and drop.
- Keyboard navigation is not implemented as touchscreen input has been prioritized.

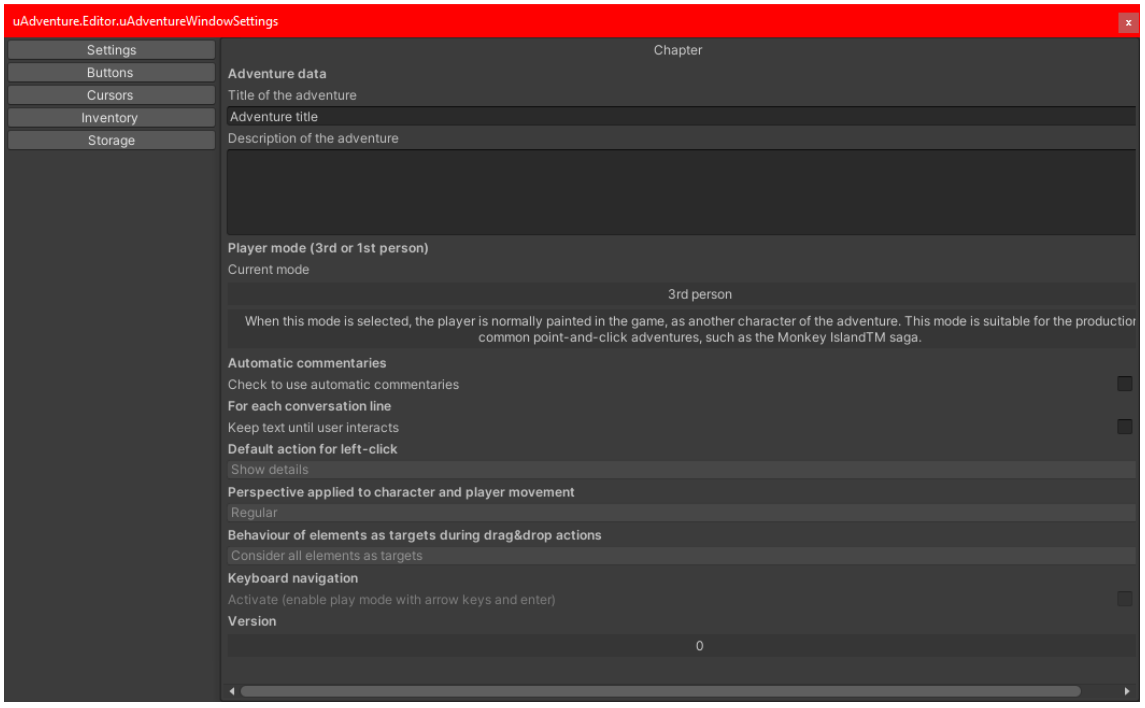


Figure 134. Info and settings window.

7.2.2. Action buttons appearance

In the buttons tab (Figure 135), the appearance of the action buttons can be changed by selecting a texture. Keep in mind that when changing a button texture, you should change its highlighted appearance texture too and both should have the same resolution to avoid visual glitches. The seven different action buttons can be configured. Custom actions, however, must be configured individually in each element.

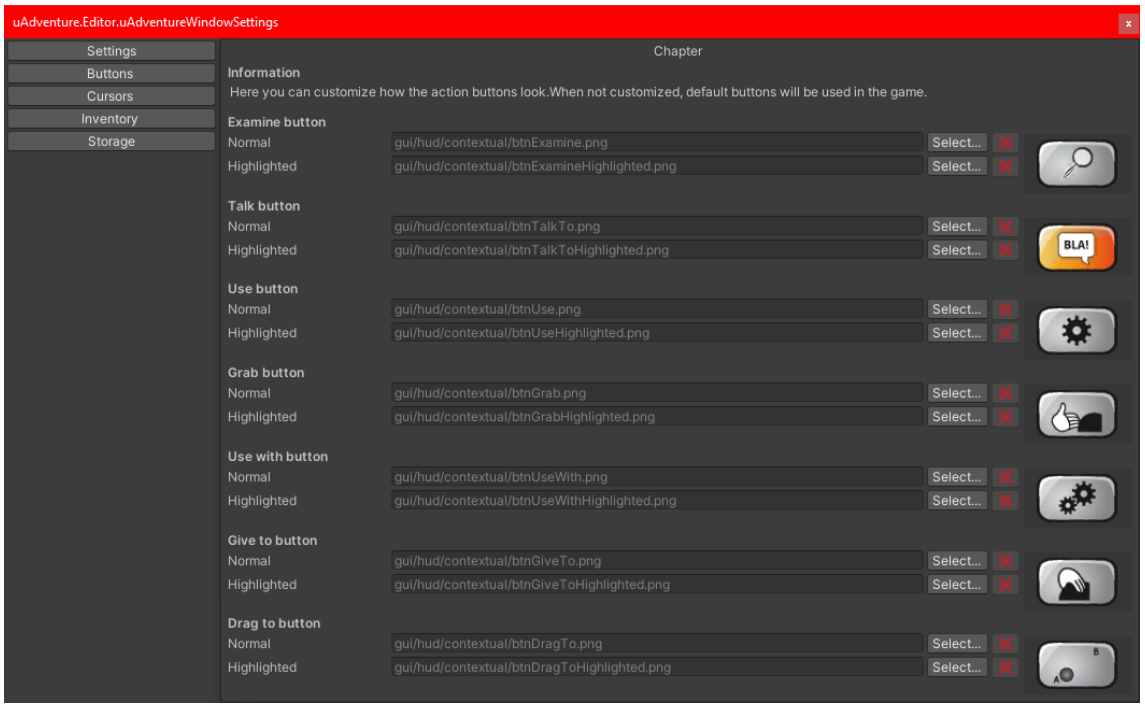


Figure 135. Action buttons configuration window.

On the right of each button you can see a preview of the look of the button. To see its highlighted view, move the mouse over the button. Please keep in mind that the image resolution is not reflected in the preview.

7.2.3. Cursors appearance

In the cursors tab (Figure 136) it is possible to change the cursors appearance. There are four different cursors: default, element, action and exit.

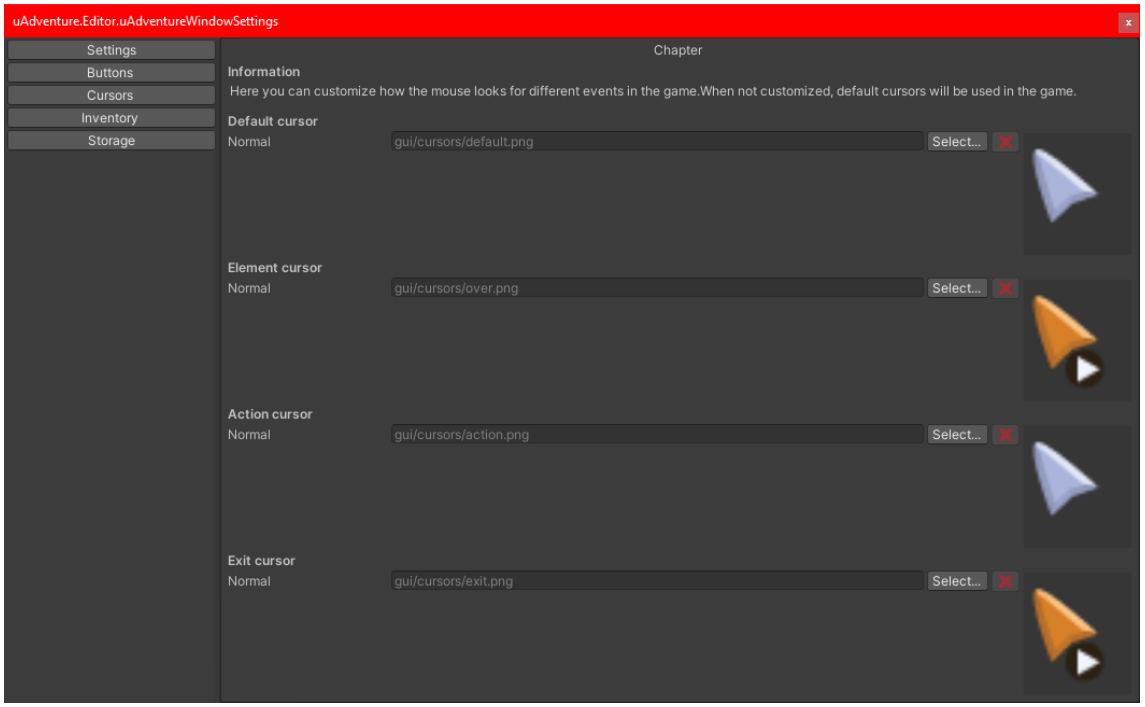


Figure 136. Cursors appearance window.

- Default cursor: is shown when the mouse is not hovering any interactive element.
- Element cursor: is shown when the mouse is hovering scene elements including items, inventory items, active areas and characters.
- Action cursor: is shown when the mouse is hovering an action or when clicking the element will trigger an action (in first-action behavior mode).
- Exit cursor: is shown when the mouse is hovering an exit.

7.2.4. Inventory configuration

In the inventory tab (Figure 137) it is possible to change the inventory display style and appearance or disable it completely. There are six different inventory configurations: top-bottom, top, bottom, fixed top, fixed bottom and icon freely positioned.

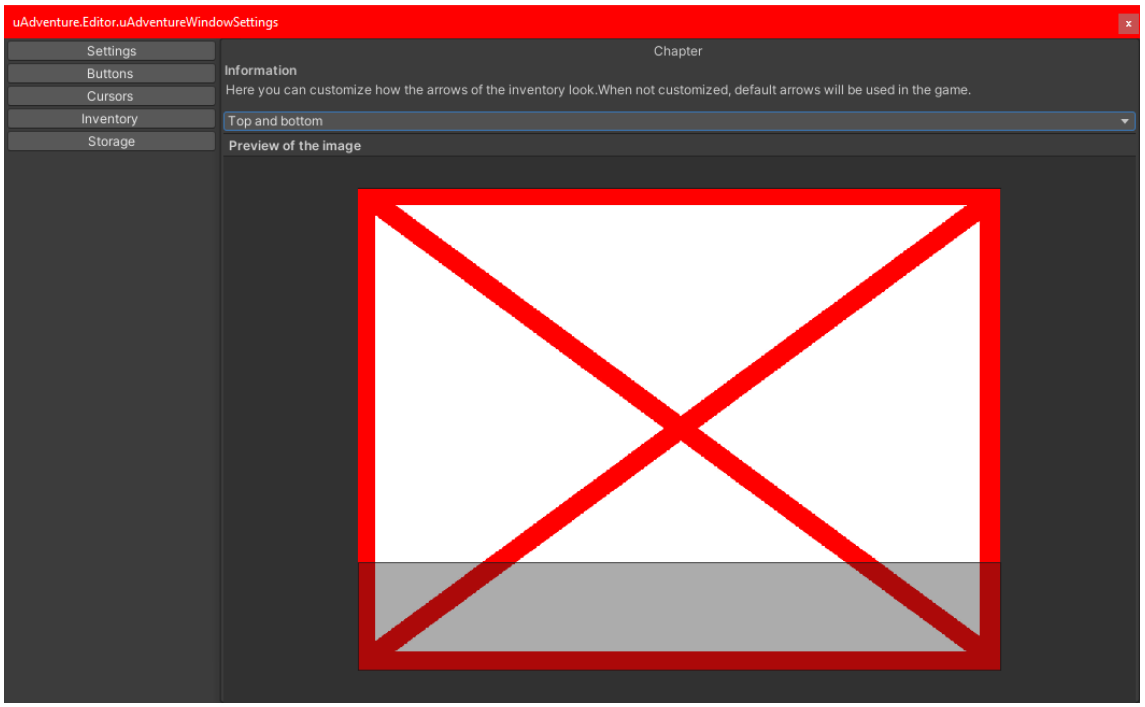


Figure 137. Inventory window displaying the inventory on the bottom.

The panel below will simulate the final appearance of the inventory according to the following description:

- Top-bottom *(eAdventure default)*: The inventory will slide when the cursor is close to the top or bottom edges, displaying the items horizontally.
- Top: The inventory will slide from the top edge displaying the items.
- Bottom: The inventory will slide from the bottom edge displaying the items.
- Fixed top: The inventory will stay opened on the top edge.
- Fixed bottom: The inventory will stay opened on the bottom edge.
- Icon freely positioned (*uAdventure default*): The inventory will be represented as an icon in the screen and its position, size and texture can be changed on the inspector window.

Despite of having different inventories, **when using handheld devices we strongly recommend**

using icon freely positioned or fixed inventories, as there is no easy way to trigger the inventory open in such devices.

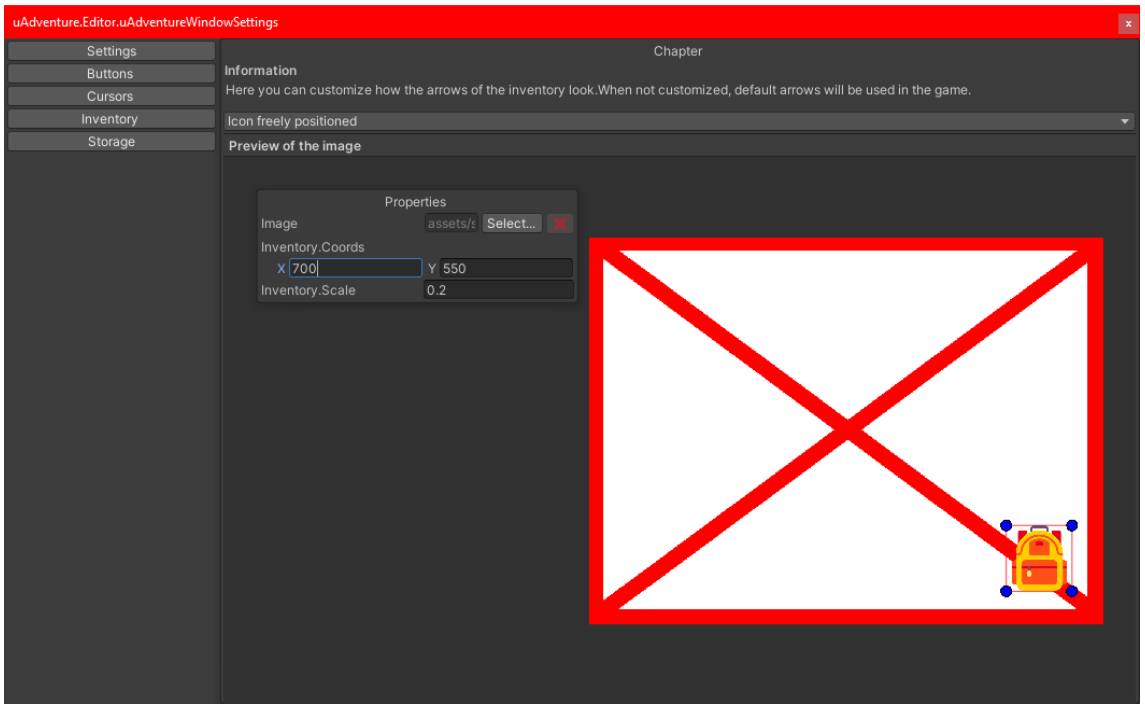


Figure 138. Inventory as an icon freely positioned. Properties such as image, cords and scale are displayed in the inspector.

To change the position and size of the icon either insert the values manually or use the handles around the icon in the preview.

7.2.5. Storage configuration

The storage configuration tab (Figure 139) allows configuring the saving and loading game behavior and menu elements. From it, it is possible to configure the different properties:

- Allow save/load: displays or hides the save and load buttons from the exit menu (pressing exit in-game will open the exit menu).
- Allow reset: displays or hides the reset button. Using the reset button will restart the game.
- Auto-save: when enabled, saves the game before and after an action or exit execution is finished. Hence, if the game is closed mid execution the game will be saved just before the action was triggered. This property is not used in editor/debug mode to simplify debugging.
- Save on suspend: when enabled saves the game when the game window loses focus or is closed. This is extremely useful in handheld devices where the screen can lock automatically after a period and the device might clear the game context in the process or the user might accidentally close the game. This property is not used in editor/debug mode.
- Restore after open: when enabled will load the last saved game automatically when the game is opened. In combination with save on suspend, in handheld devices can be used to continue the gameplay right away if the game is closed. This property is not used in editor/debug mode.

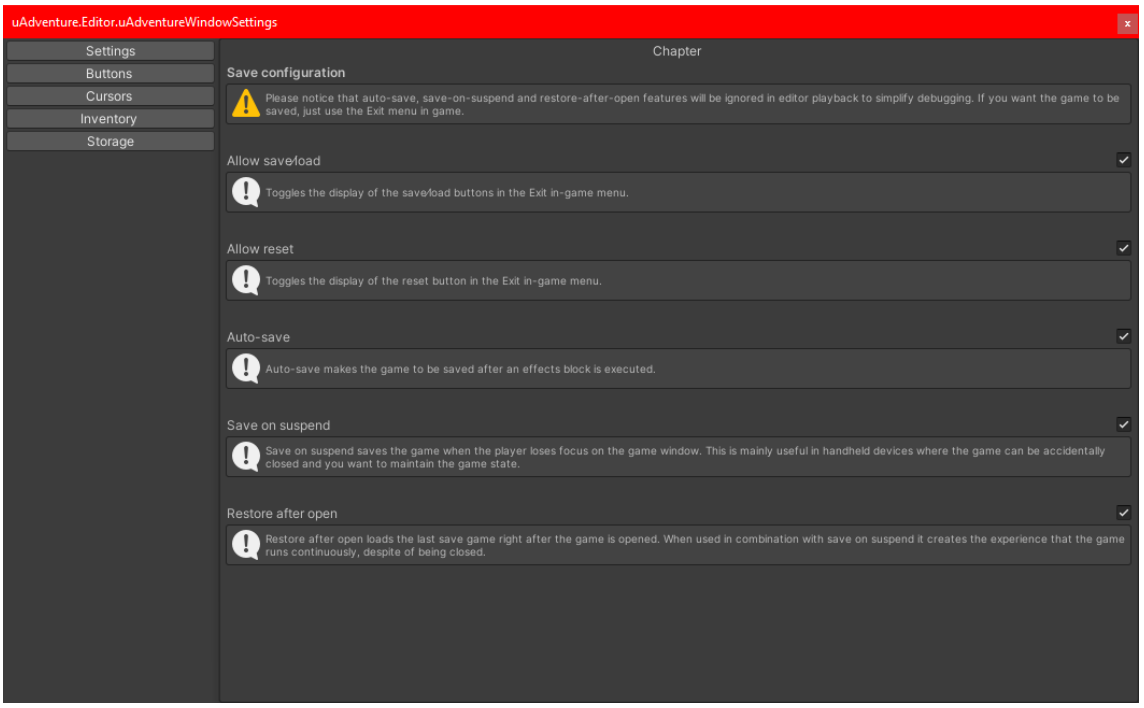


Figure 139. Storage configuration window.

7.3. Chapters menu

This menu can be used to add, delete, import and order chapters. In addition, flags and variables of the chapter can be edited using the option included in this menu.

7.4. Configuration menu

This menu allows configuring the language of the editor. Currently <u-Adventure> is available in Spanish, English, Romanian, German and Portuguese. Please visit http://u-Adventure.e-ucm.es/contributors/ to see a full list of people that kindly produced each translation. Without them <u-Adventure> would never be multi-language. Thanks!