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,
Manuel Freire Morán,
Iván Martínez Ortíz,
Baltasar Fernández Manjón

Latest version: 1.0

Updated: Jul 2019

License

Acknowledgements

<u-Adventure> version 0.2 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 S2013/ICE-2715.

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 2017.3. 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 tutorials and other learning materials on our website (http://u-Adventure.e-ucm.es). You can also access the

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

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. 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 1.4.3) 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

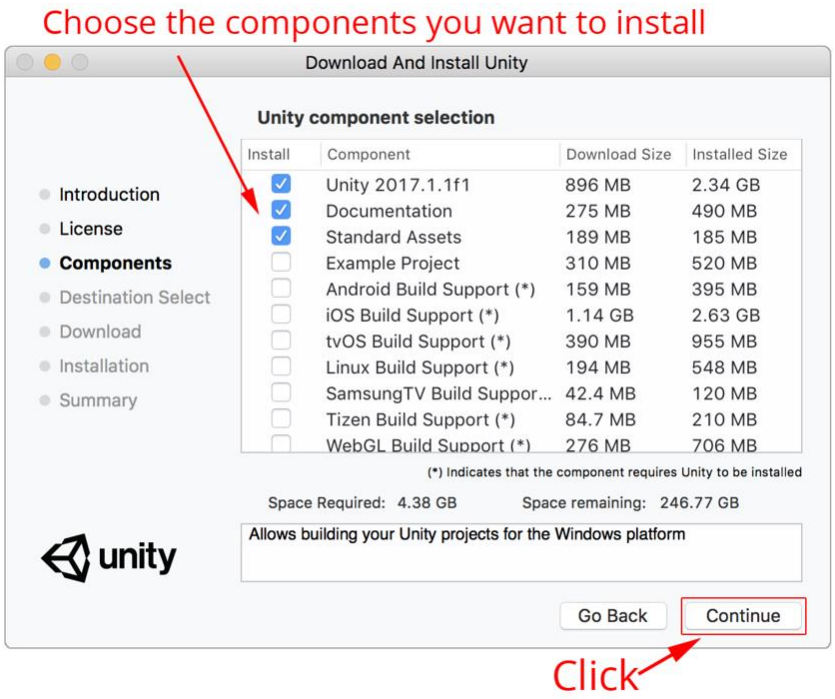

Figure 1. Unity installer platform selection. From Unity3d.com

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.

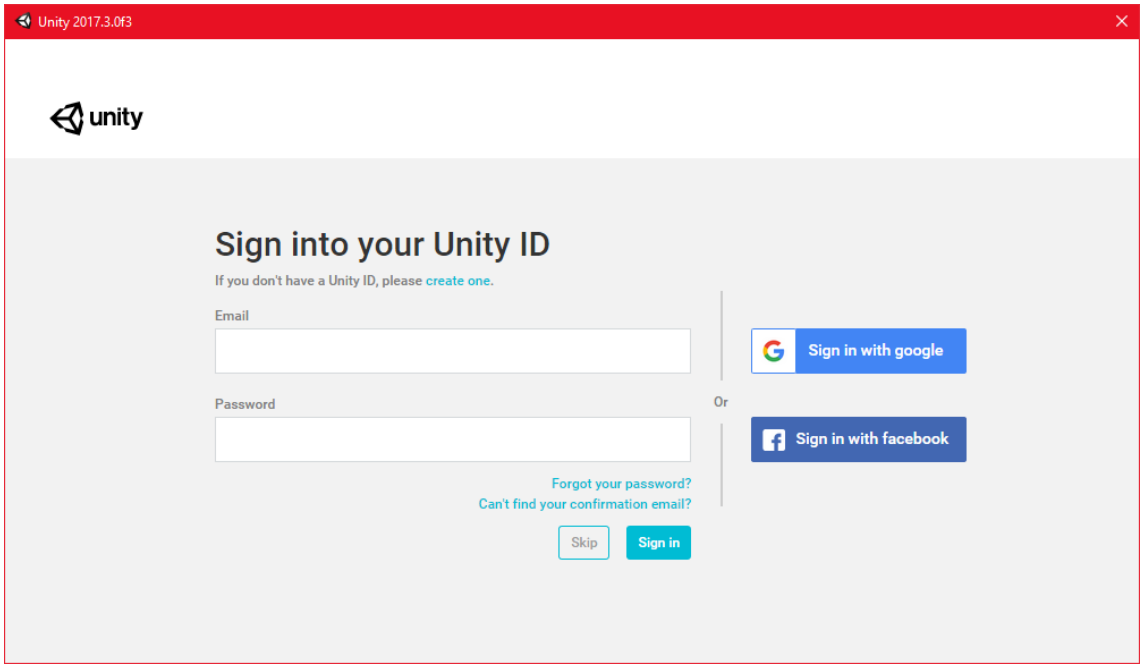

Figure 2. Unity log-in screen.

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

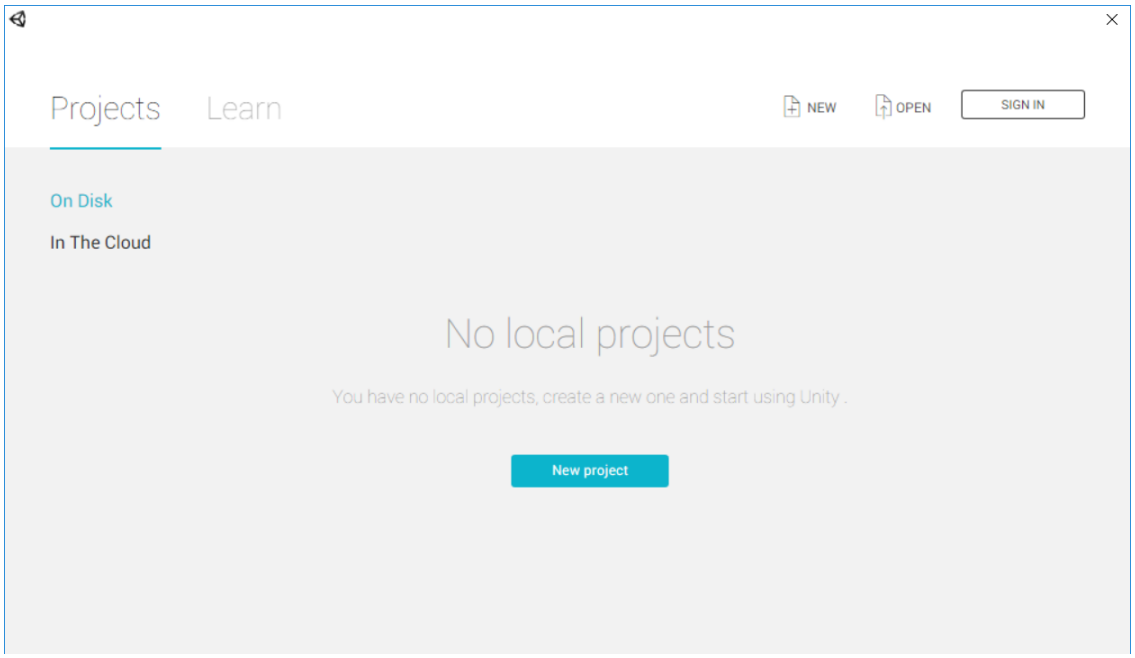

Figure 3. Unity project management window

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.

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 unless you specifically want to use Unity analytics. More information about Unity Analytics can be found at the webpage[3].

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 4. 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

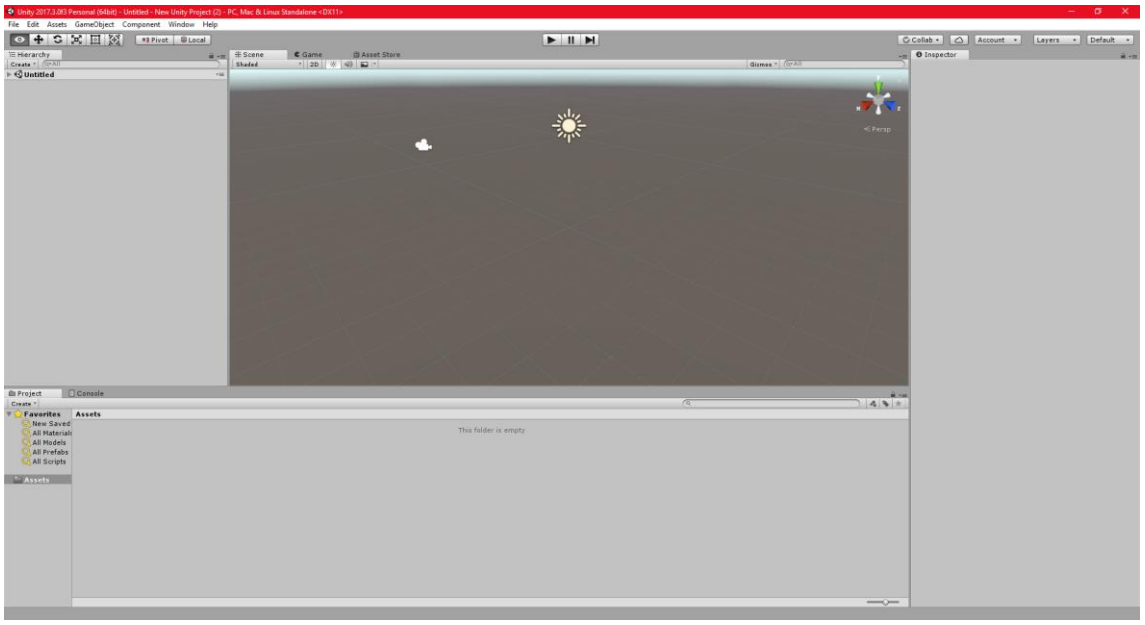

Figure 4. Unity editor with default layout.

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]). The file is identified by its extension "unitypackage" and, once downloaded, the Unity logo will appear as the file icon.

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 5). 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 5).

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

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

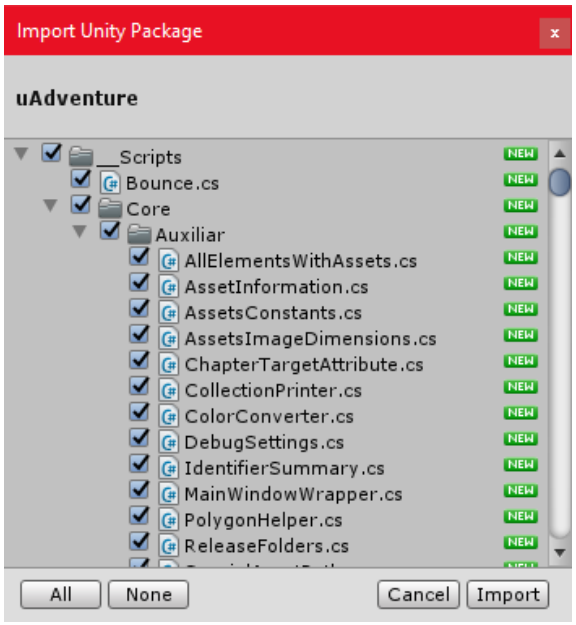

Figure 5. Unity package import window.

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.

1.3.3.2 Installing through the Unity Assets Store

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

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" 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*^{TM} 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*^{TM} 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 6. uAdventure welcome window.

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

1.4.2. 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. By opening a project all the previous project data is removed, including both model and assets!

1.4.3. 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 a ".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 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 an index.html and three folders (Build, StreamingAssets and TemplateData). 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. If you want to install a new build platform 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 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 contrasts to previous <e-Adventure> games where the items were interacted 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. Press on any of the options of the menu to trigger the interaction.

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

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

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

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

Figure 7. 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.

2. My first <u-Adventure> game

Once a Unity project with uAdventure is opened, the uAdventure menu will appear on the top menu bar. We strongly recommend for previous <e-Adventure> users to use the pre-configured uAdventure layout to simplify the uAdventure usage and avoid messing around with the Unity windows. To use the uAdventure layout just click in the top menu in "uAdventure > Configure Layout". This way the Unity editor will restart and show the uAdventure main editor. Otherwise, you can just open the uAdventure editor window at "uAdventure > Open uAdventure Editor".

We will start by studying the options available in the editor (Figure 8). 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 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.

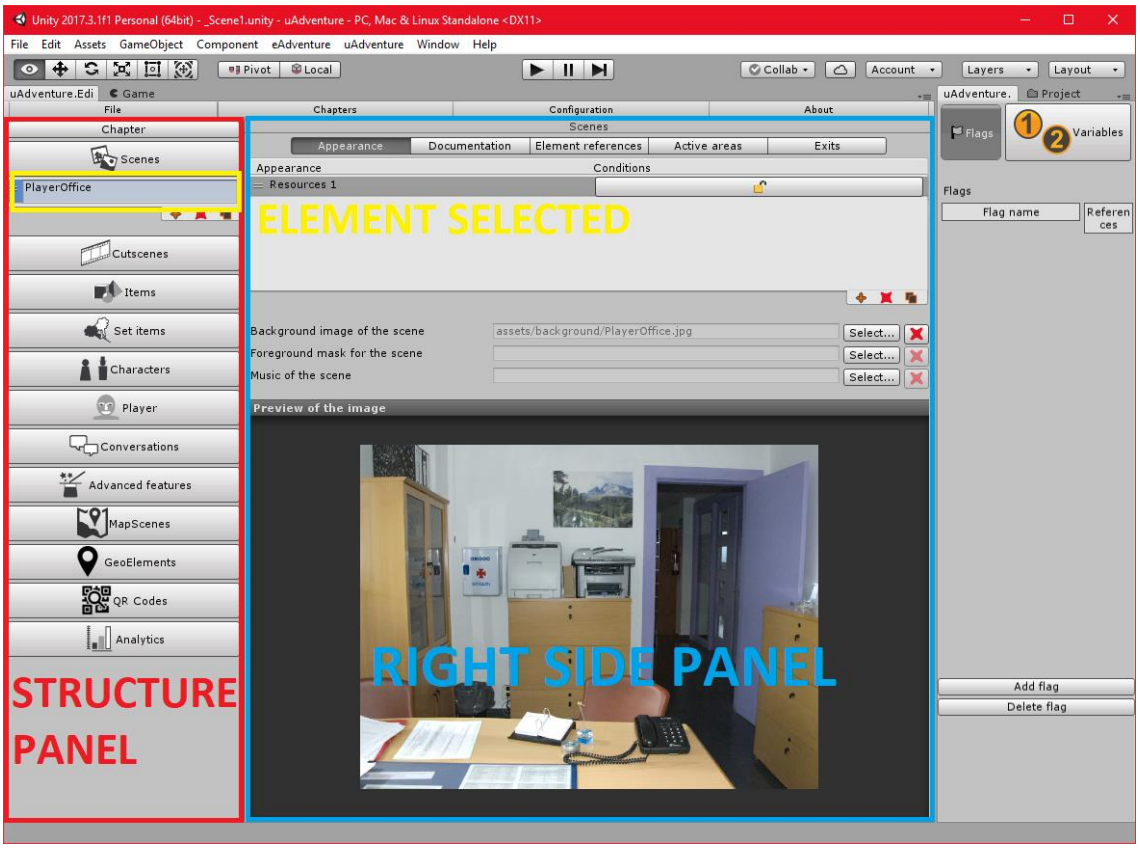

Figure 8. Initial view of the <u-Adventure> editor, once a new project is created or an existing one opened.

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 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 menu-bar) has options to create, import, delete and organize the chapters in the game, as well as to modify of the current chapter's name. This menu also has options to adjust the *flags* and *variables* available in the chapter, but the chapter flags and variables menu will appear on the right side in the default menu. The flags and variables will be detailed later.

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

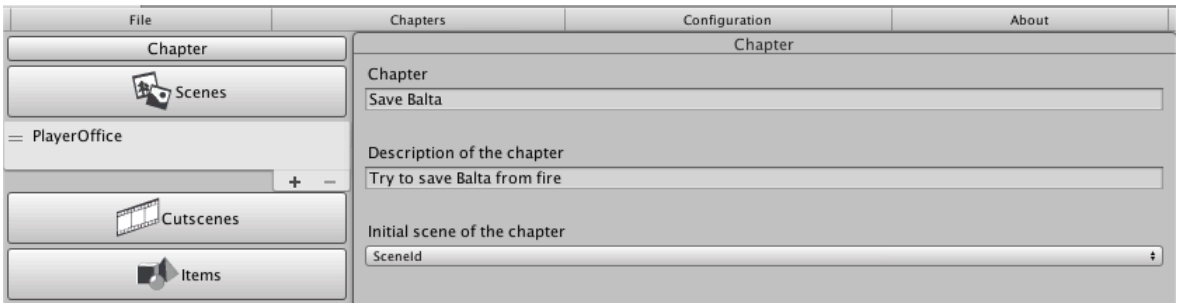

Figure 9. Chapter edition.

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, that is, where the player interacts with the objects and characters in the game, and these are connected to other scenes by *exits* that we will use later. On the other hand, *cut-scenes* are used to increase the educational value of games and can be either *Slidescenes* or *Videoscenes*. *Slidescenes* are made of slides or images that are shown using the full screen. *Videoscenes* will display a video using also 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 (Figure 10). 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, that 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>.**

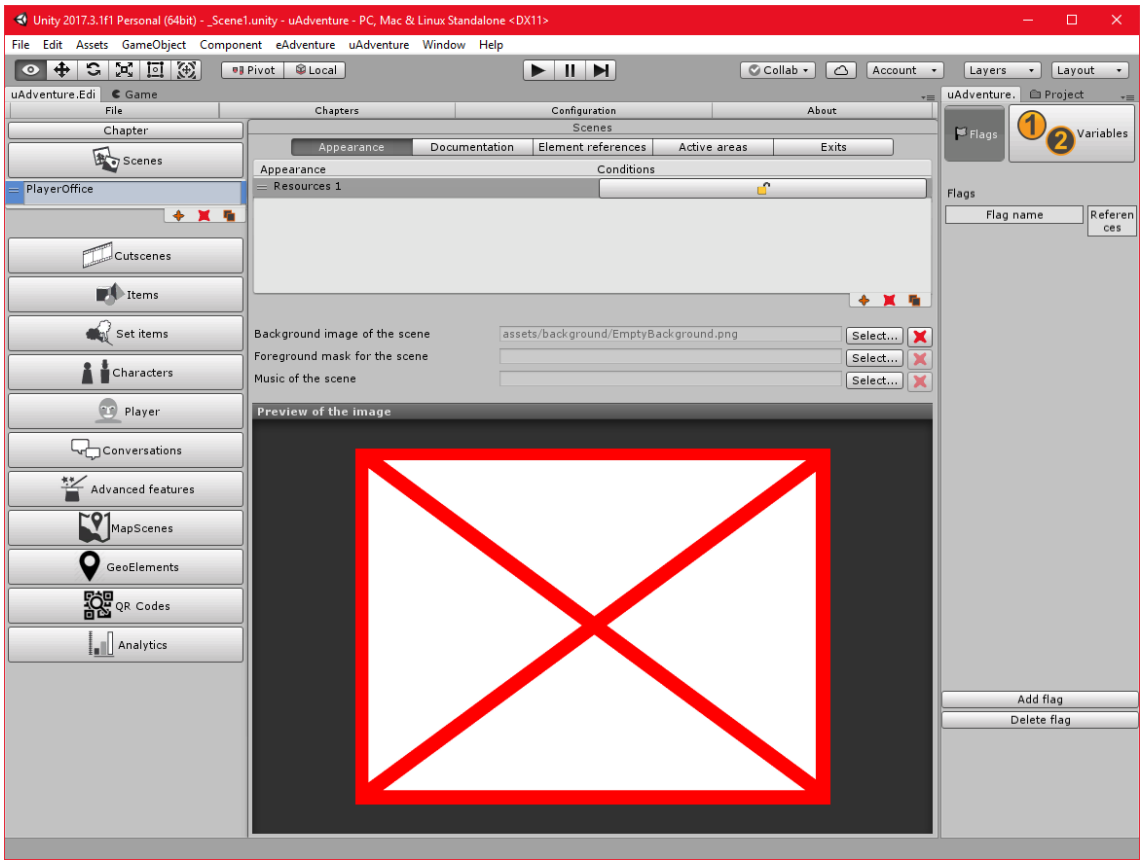

Figure 10. Scene edition shows a preview of all the scenes in the chapter. By default, games include an empty scene.

When a scene is selected (Figure 11) 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.

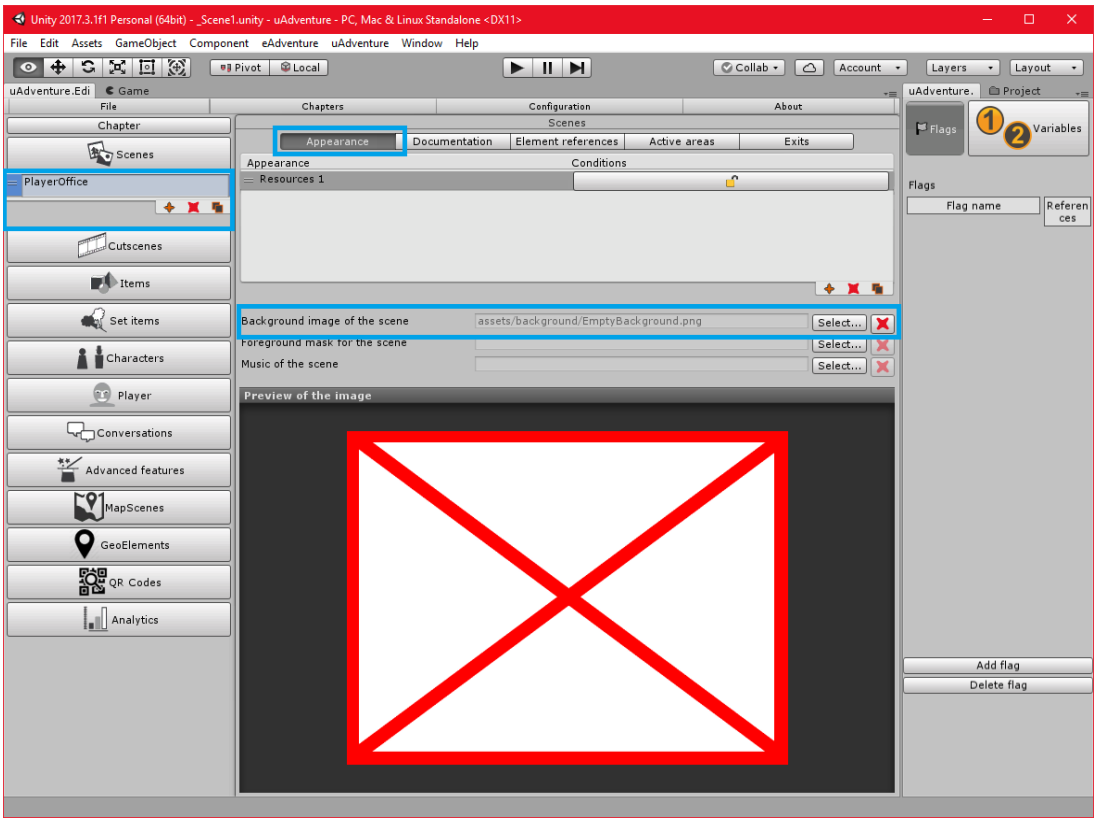

Figure 11. Scene edition panel, with the "Appearance" tab selected.

The “Appearance” tab is used to change how the scene is shown (or how it is perceived), and the same is valid 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 one (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 sceneand 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 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.

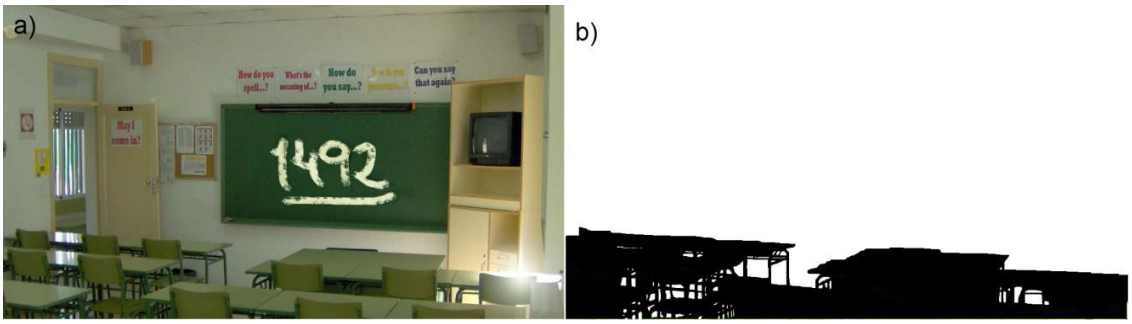

Figure 12. Background image (a) and mask (b). The back parts of the mask mark the parts of the background image that will be painted in the front of other elements and the white parts those that will be painted behind.

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*. 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 asset. Clicking this will display a dialog to select an image (Figure 13).

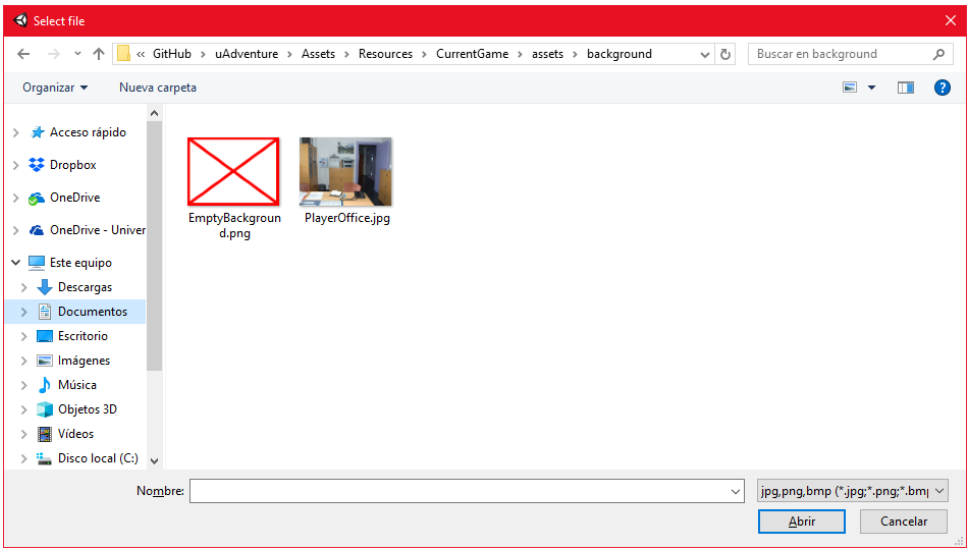

Figure 13. Assets selection dialog for the scene background.

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

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

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).

Since this game is a third person game, foreground masks do not make sense since there is no player. After these steps are completed we can see the result in Figure 14.

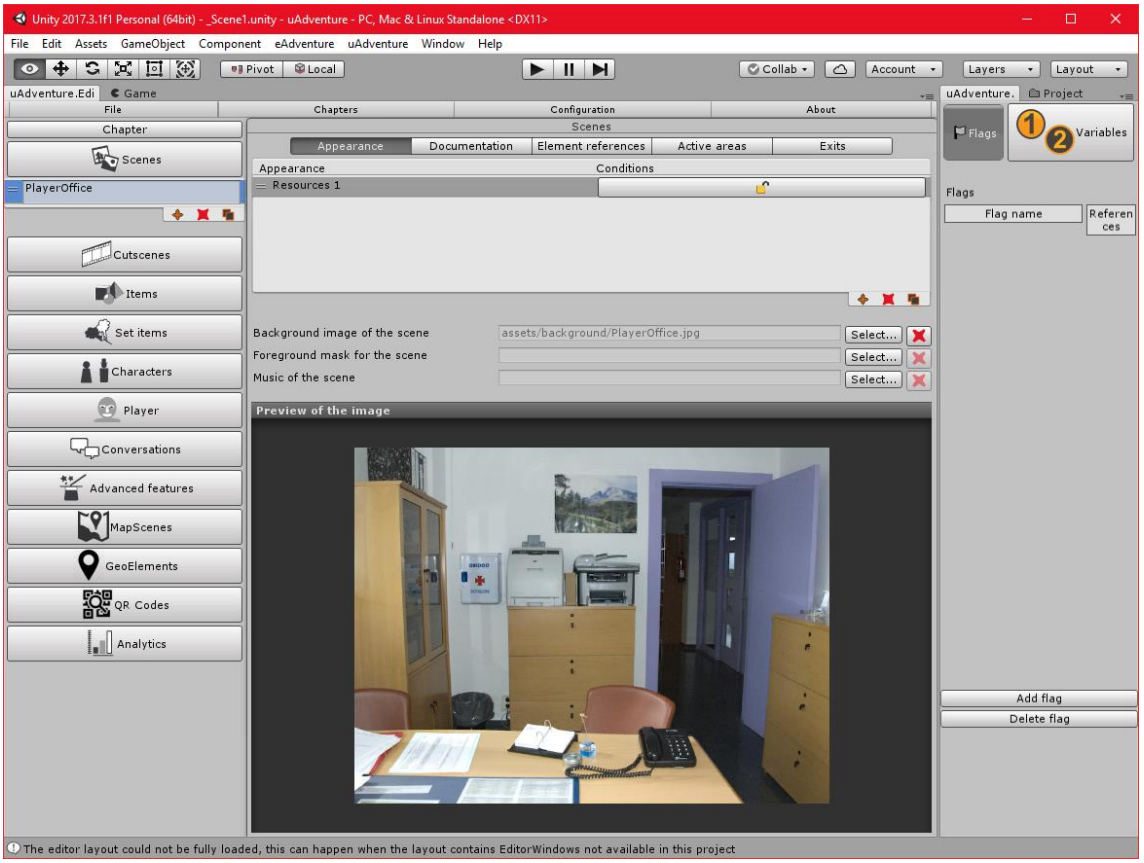

Figure 14. Resulting *PlayerOffice* Scene.

Before the previously modified scene is shown to the player, it is possible to add a cutscene that will present the game. In this case, the cutscene 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 thought of as invisible rectangles or shapes. When the player clicks on an exit, this results in a change of scene (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 14 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 15).

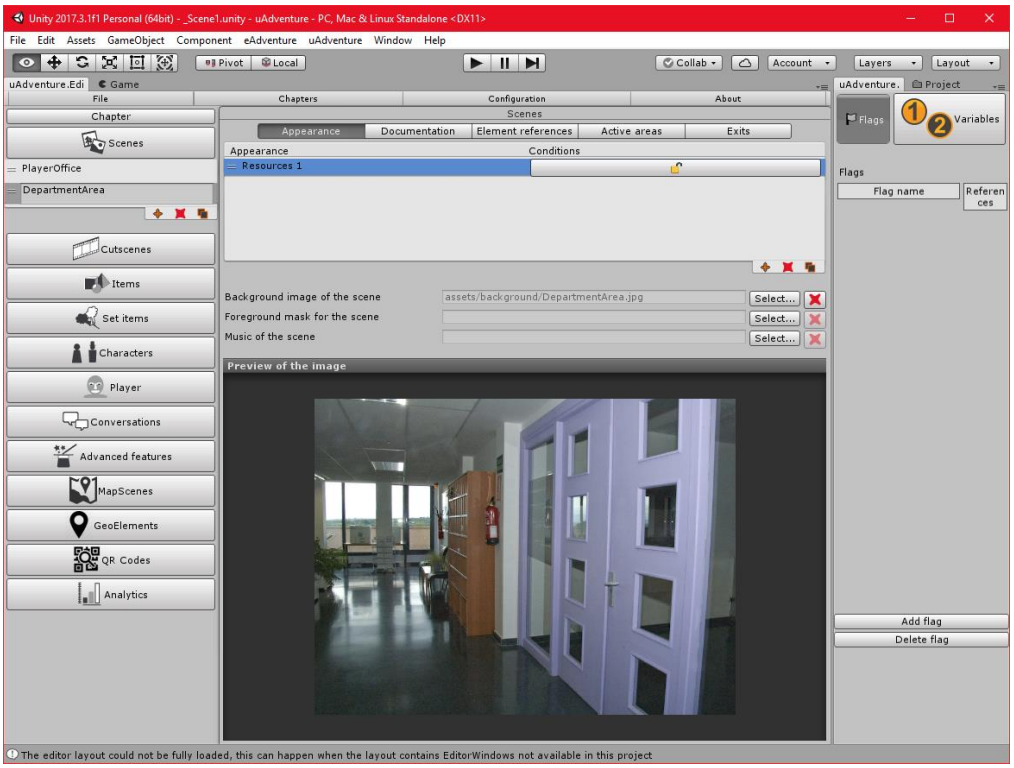

Figure 15. Department Area scene.

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 we click on it, we see a new red rectangle in the scene and a new line in the top list. To change the destination, click on the element in the list or in the scene and select the destination in the list.

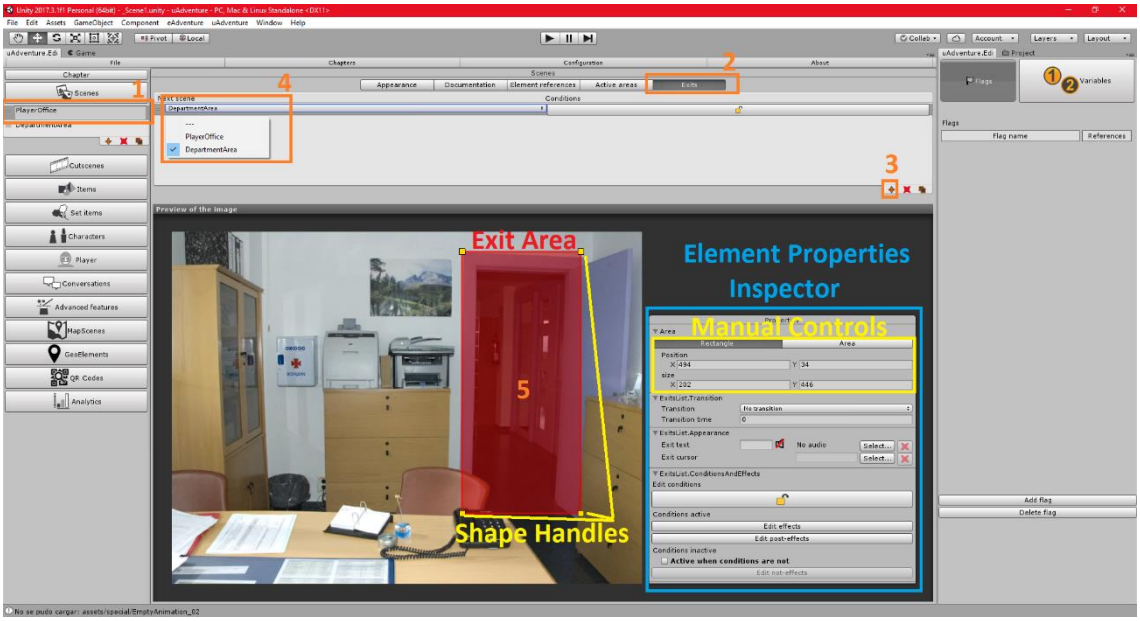

Figure 16. Creation and configuration of an exit. Steps are 1), select the scene, 2) select the exit tab, 3) press the + button, 4) select the element and pick a destination and 5) define the area.

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 17).

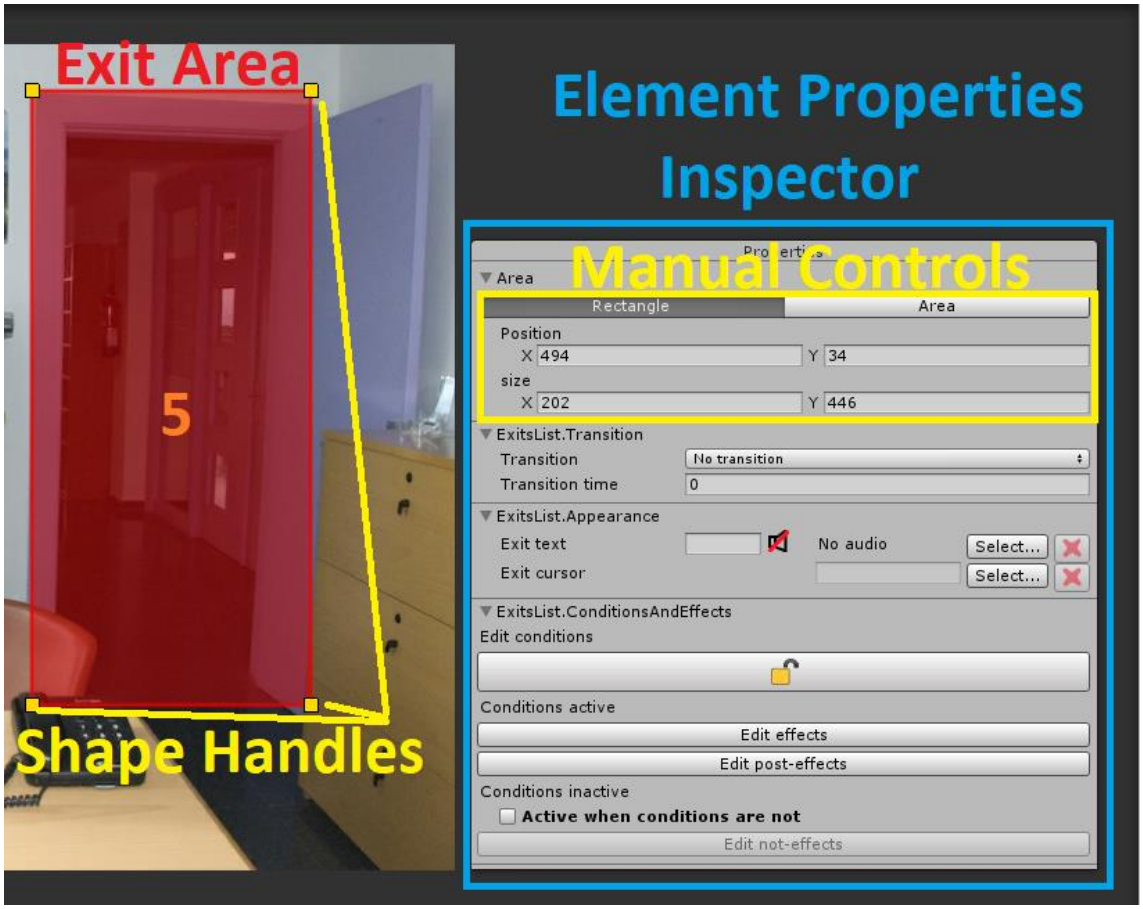

Figure 17. Element properties inspector and area shape.

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 as seen in Figure 18. When in the "Area" mode, the handlers become blue and round to easily identify the mode.

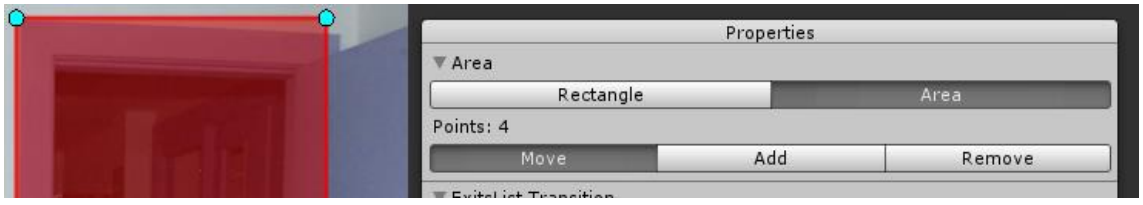

Figure 18. Area shape mode showing the different control tools and the blue roundly handlers.

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. This view can be used to identify 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.

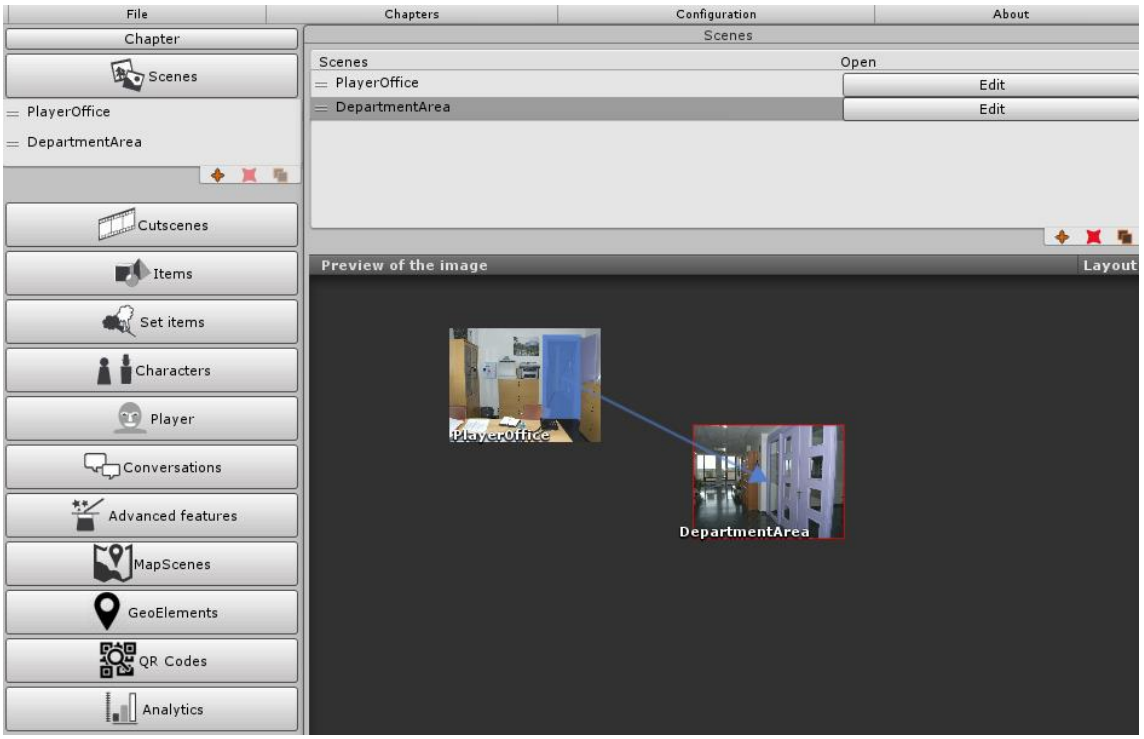

Figure 19. Main scene preview showing different exits.

Now you can run your game again and see how you can change to the other scene by clicking in the "invisible" area that is located on the door frame (Figure 20).

Figure 20. 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

Before being able to test the game, the player assets must be defined; this is studied in detail in section 2.8. However, it is possible that the player would still appear 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 (Figure 18). 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.

2.3. Cutscenes

<u-Adventure> cutscenes 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 cutscenes. There are two basic types: Slidescenes and Videoscenes. Slidescenes are made up of a series of images. Although Videoscenes 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[9].

2.3.1. Slidescenes

To add a new slidescene,select the Cutscene left menu and click the add button. When clicked, two options appear for cutscenes and slidescenes (Figure 21). Choose "Add slidescene" to continue. Selecting the slidescene it will result in a new panel with three tabs in the right panel. (Figure 22).

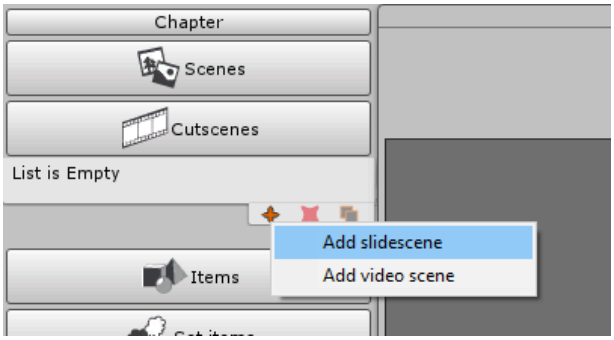

Figure 21. Create cutscene options.

The three tabs in the slidescene edition panel are:

- "Appearance": this is used to set up the images that make up the slides of the cutscene. Slidescenes use animations, that 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.
- "Cutscene end configuration": this tab is used to establish what happens when the cutscene ends, see 2.7.3 for details.

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

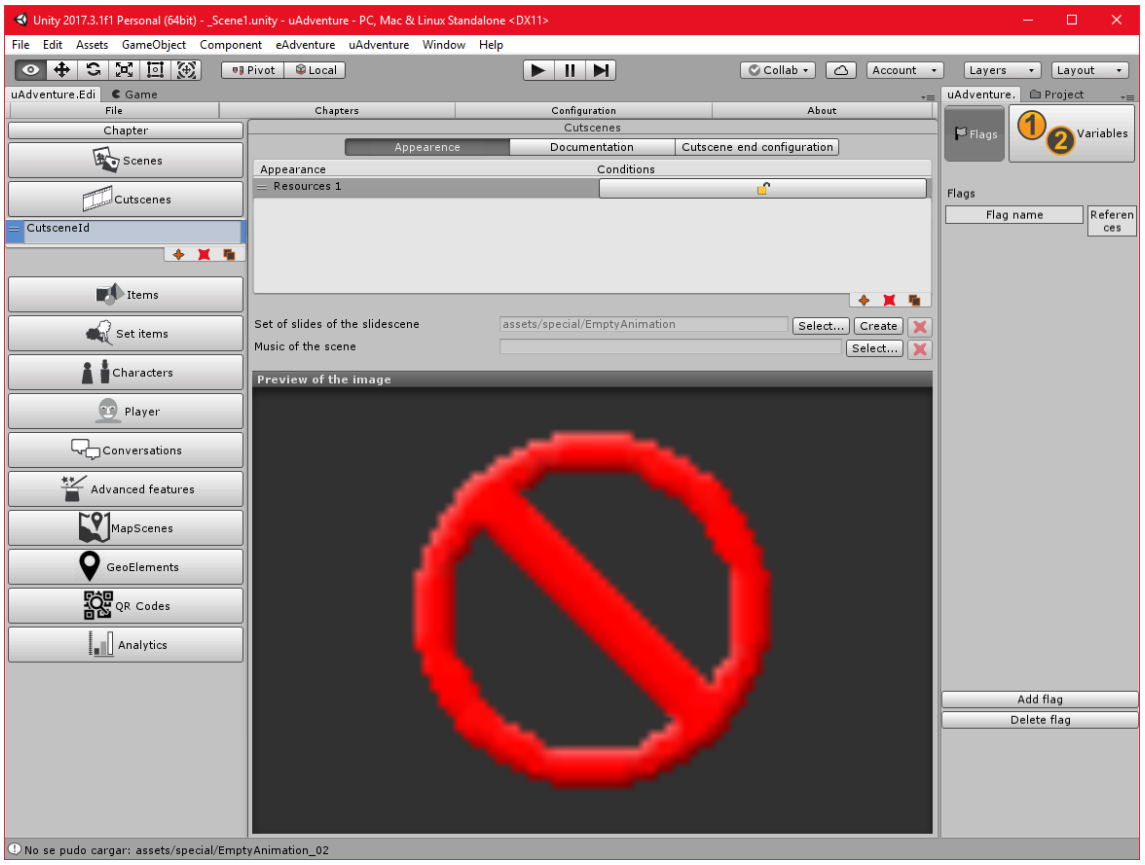

Figure 22. Slidescenes tabs.

2.3.2. Videoscenes

Videoscenes are added in a similar way to slidescenes. The tabs in the videoscene edition panel are the same that are found in the slidescene 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 cutscenes with other scenes

Just as regular scenes, cutscenes can be connected to other scenes in the game. The process is similar to the one used for regular scenes; the difference is that cutscenes only have one exit. This is configured in the "Cutscene end configuration" by choosing one of the available options (Figure 26):

- Returns to the previous scene: if selected, when the cutscene ends the game will go back to the previous scene (either regular or cutscene). If the cutscene 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 cutscene 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 slidescene.

The purpose of a slidescene is to dynamically present noninteractive information. In contrast to the normal playstyle, in an slidescene (or a videoscene), the player cannot make decisions or use its character and inventory. Therefore, slidescenes (and videoscenes) are perfect to introduce contents or make artistic cutscenes.

We are going to use a slidescene 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.

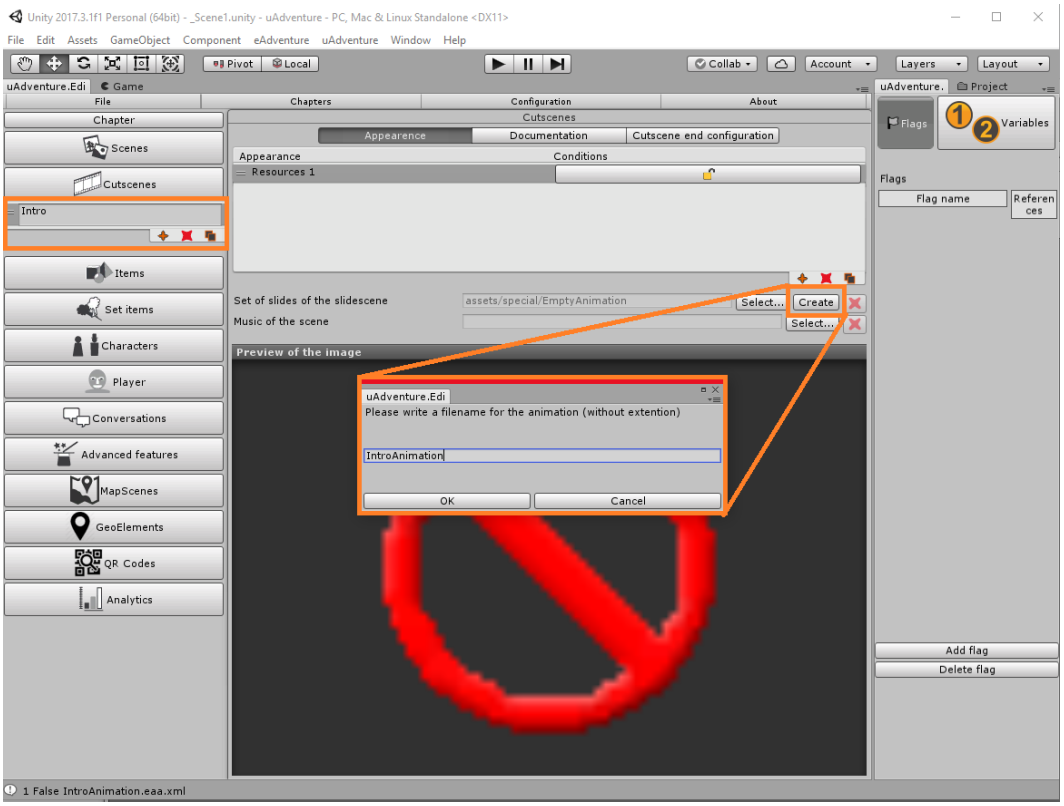

Figure 23. Slidescene animation creation process.

We can start creating a new Slidescene 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 slidescene is still using the default empty animation. To create a new animation for our slidescene we are going to click on the "Create" button in the "set of slides" field. If we already had an animation tore-use, click in the "Select…" button. After clicking the "Create" button, a prompt asking for a name appears (Figure 23).

Right after the animation is created, the Animation Editor (Figure 24) is prompted.

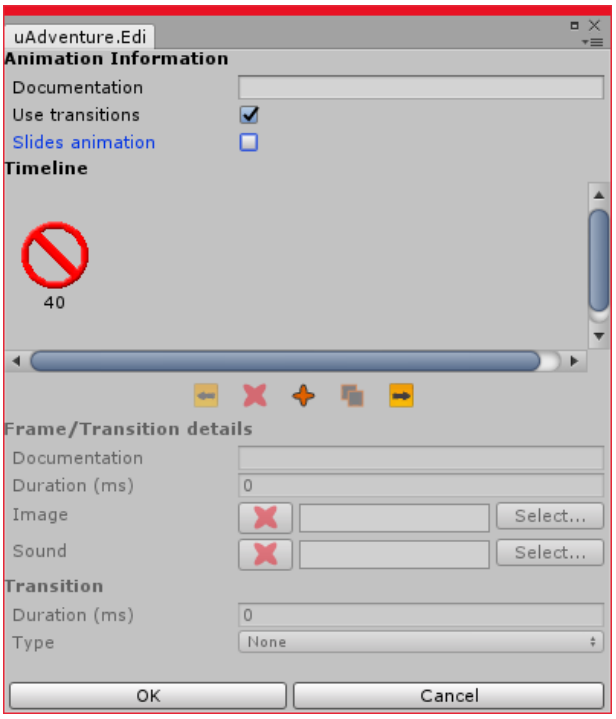

Figure 24. Animation Editor with a single empty frame.

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 cutscene we are going to disable transitions in the top checkbox. When you have selected the image for the second frame (Figure 25) click on "OK" for the animation to be written.

Figure 25. Finished animation frames.

Now, we need to tell the Cutscene to go to the Scene PlayerOffice once it is ended. To do that, switch to the "Cutscene end configuration" tab and select "Goes to a new scene" where you can choose the "Next scene" and set the value to "PlayerOffice" (Figure 26).

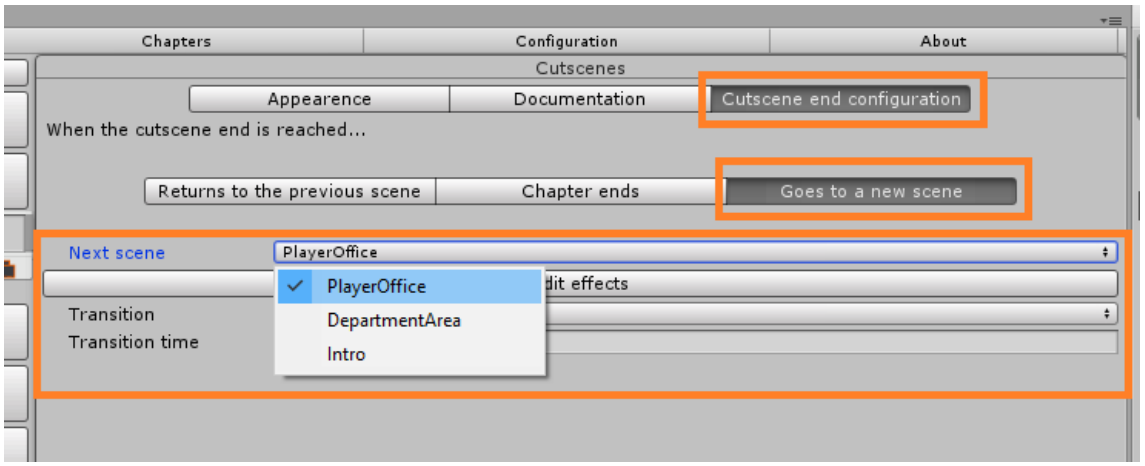

Figure 26. Cutscene end configuration to go to a new scene.

Going back to where we started, we created this slidescene to be the Intro of our game. This means, 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 27).

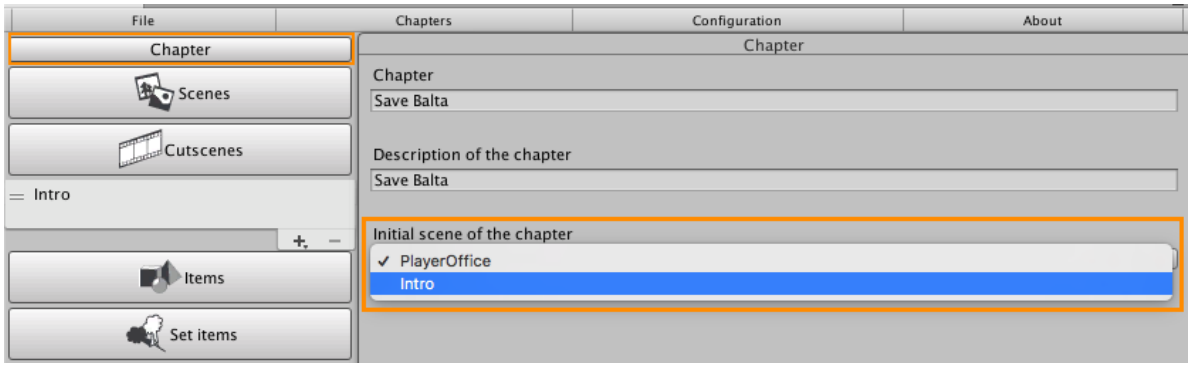

Figure 27. Chapter initial scene selection.

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 Cutscene. You also know how to control the starting point of the game and how to change from a cutscene to a scene.

2.4. Items

This section describes how to add new items (or objects) with which the player can interact with in <e-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 28). 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[10]. 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.

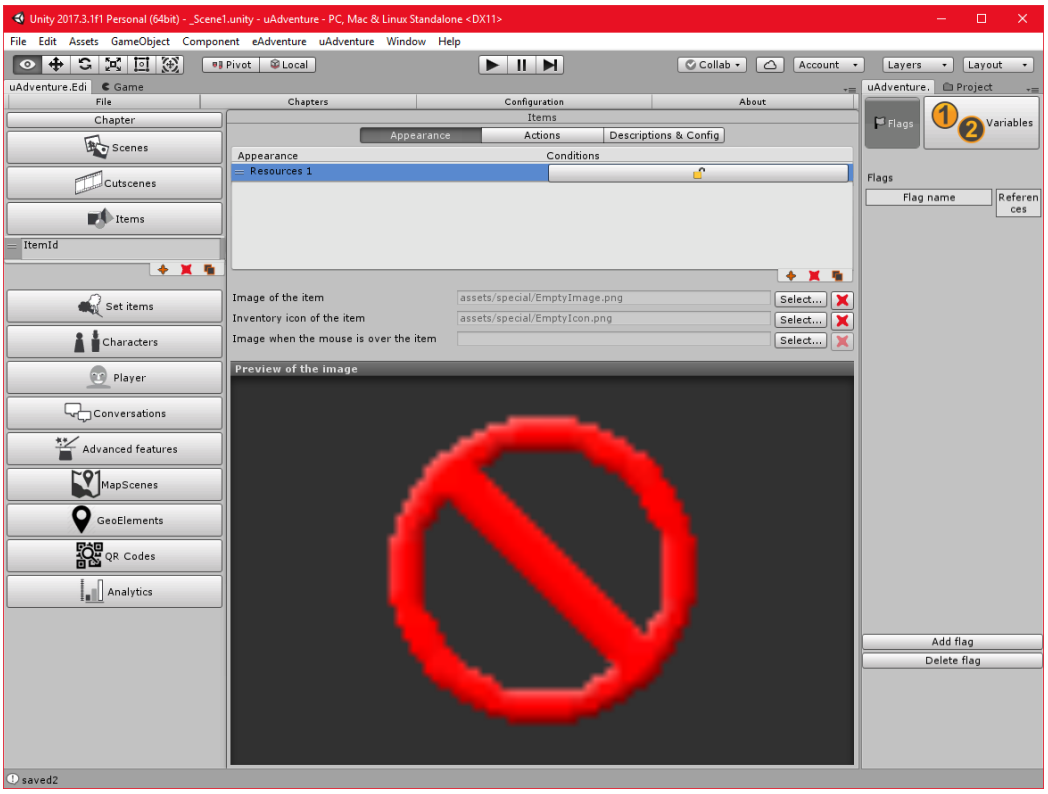

Figure 28. Items panel with appearance tab selected.

[10] 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 what are the consequences (the results) of that 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, that 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 29) are:

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 can be overridden using a special effect later described.

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 3.1.9).

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.

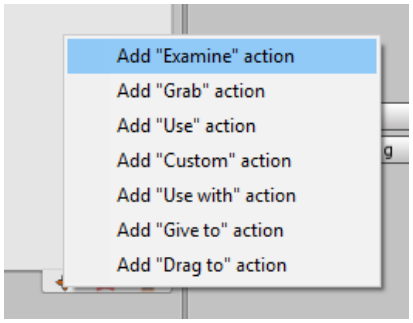

Figure 29. Possible actions.

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 valid 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.

2.4.3. Information

Just like for scenes, the information about items can be modified in the "Information" tab (Figure 30).

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

- 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.
- The item returns to its initial position: 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.

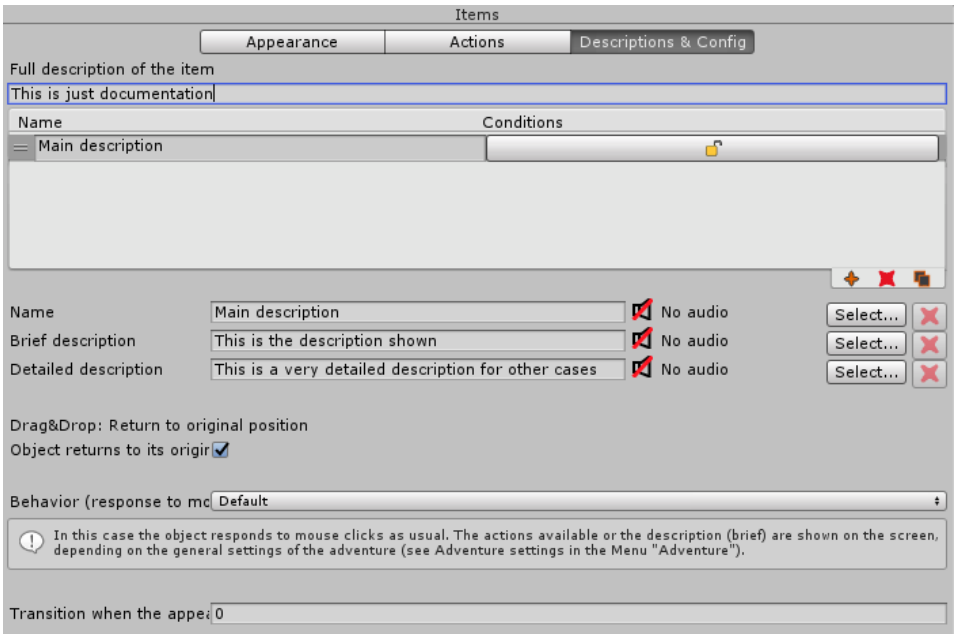

Figure 30. Items "Description and Config" tab.

2.4.4. 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. When configured. 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. Select the idle phone for the second resource pack and select the “Resources 1” back as default.

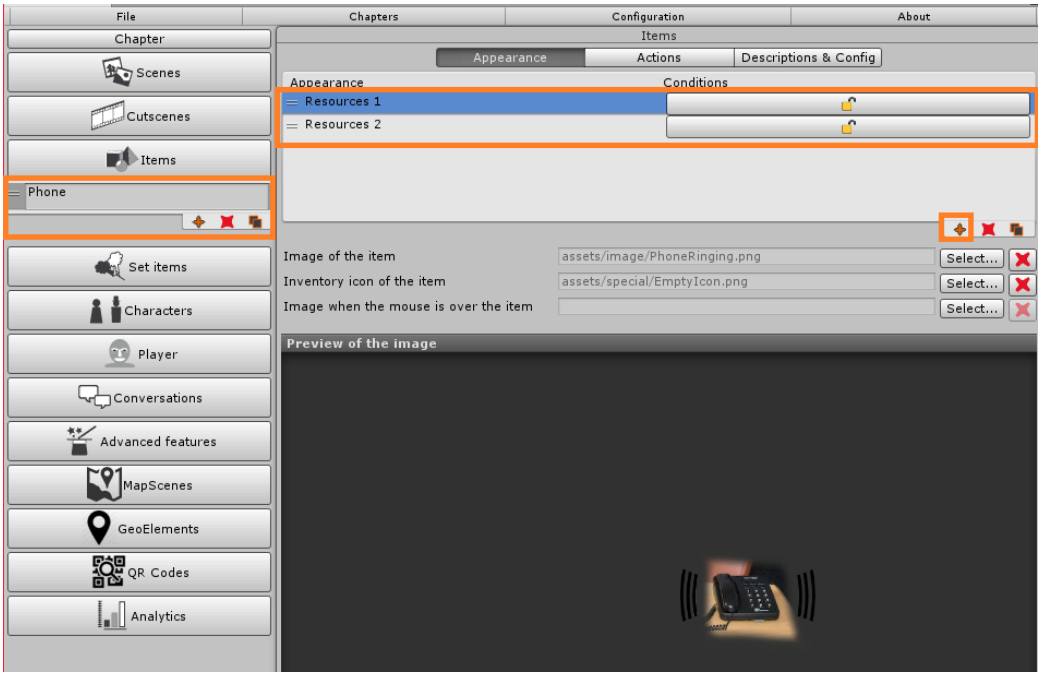

Figure 31. Phone item with two different resources for ringing and idle states.

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 to 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. However, we will explain conversations later in this guide, so we can create another slidescene instead by 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 26), to get back to the office as the phone ends. The frames used will be the ones in Figure 32.

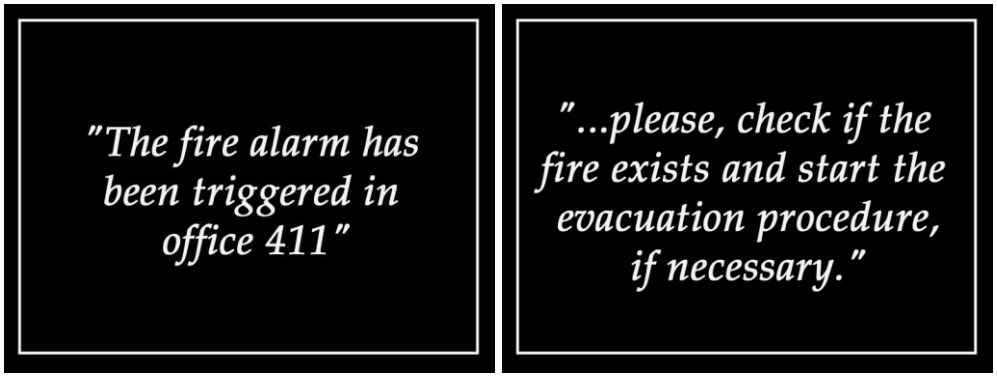

Figure 32. Phone call frames.

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.

2.4.5. 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 33).

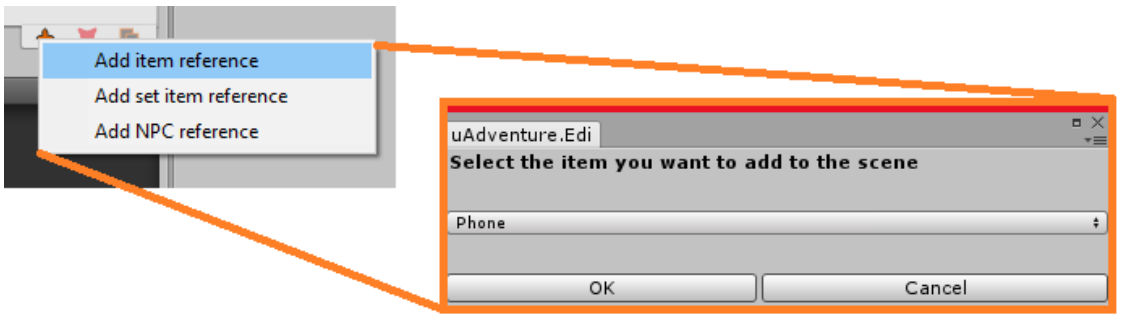

Figure 33. Add item prompt.

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 33). 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 34).

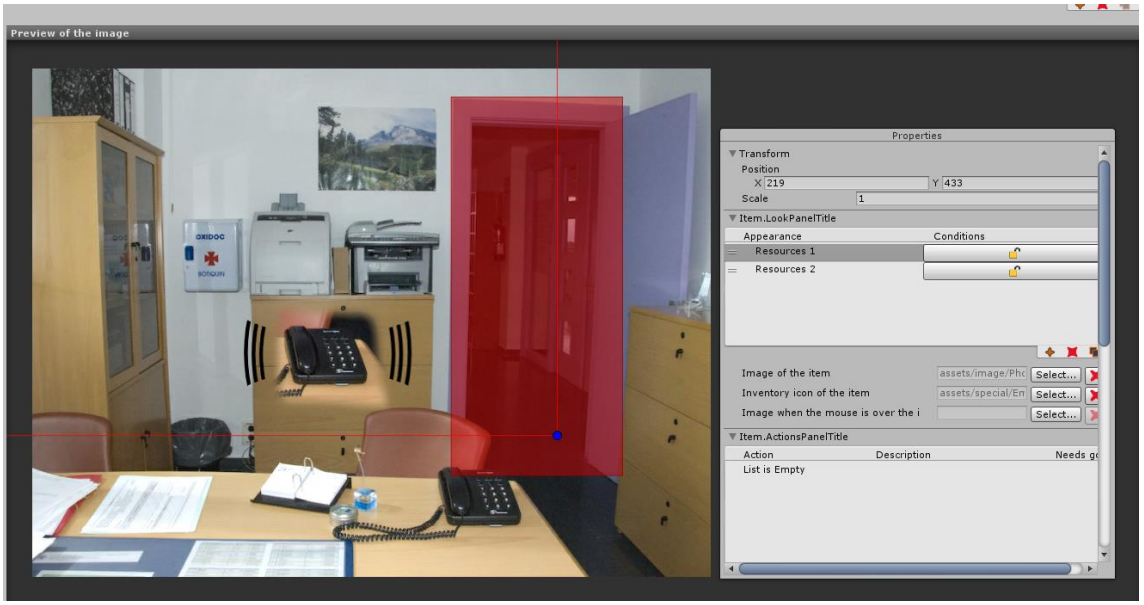

Figure 34. Scene edition panel, “Element references” tab just after adding a new element reference.

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 25).

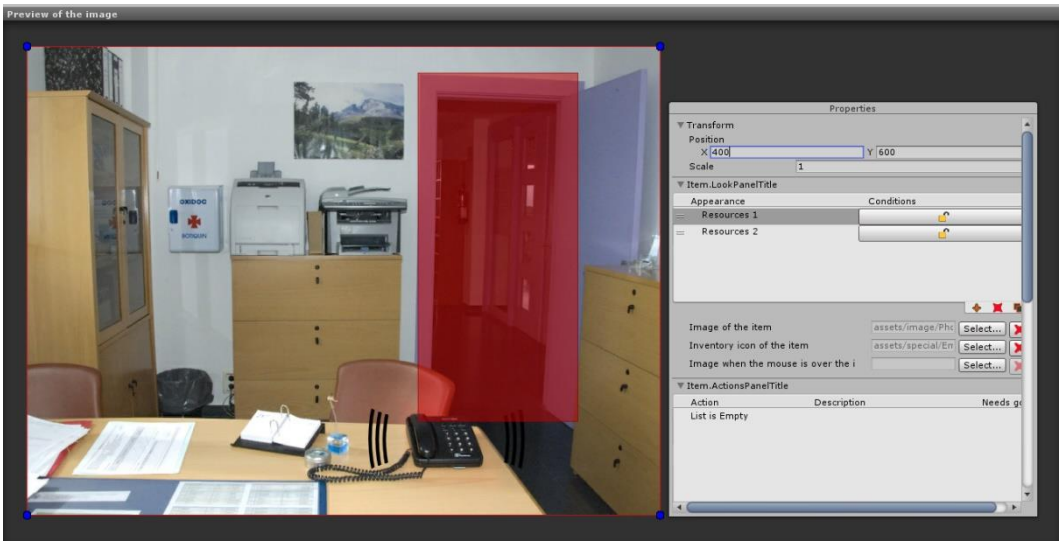

Figure 35. Scene edition panel at the “Element references” tab. After moving and scaling the “Phone”, it now sits correctly over the table in the scene.

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 XXX).

- Checkbox: This box is used to hide or show the reference in the preview. It has no effect in the game, as seen in section 2.2.4.
- 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 3.8 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 36). 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.).

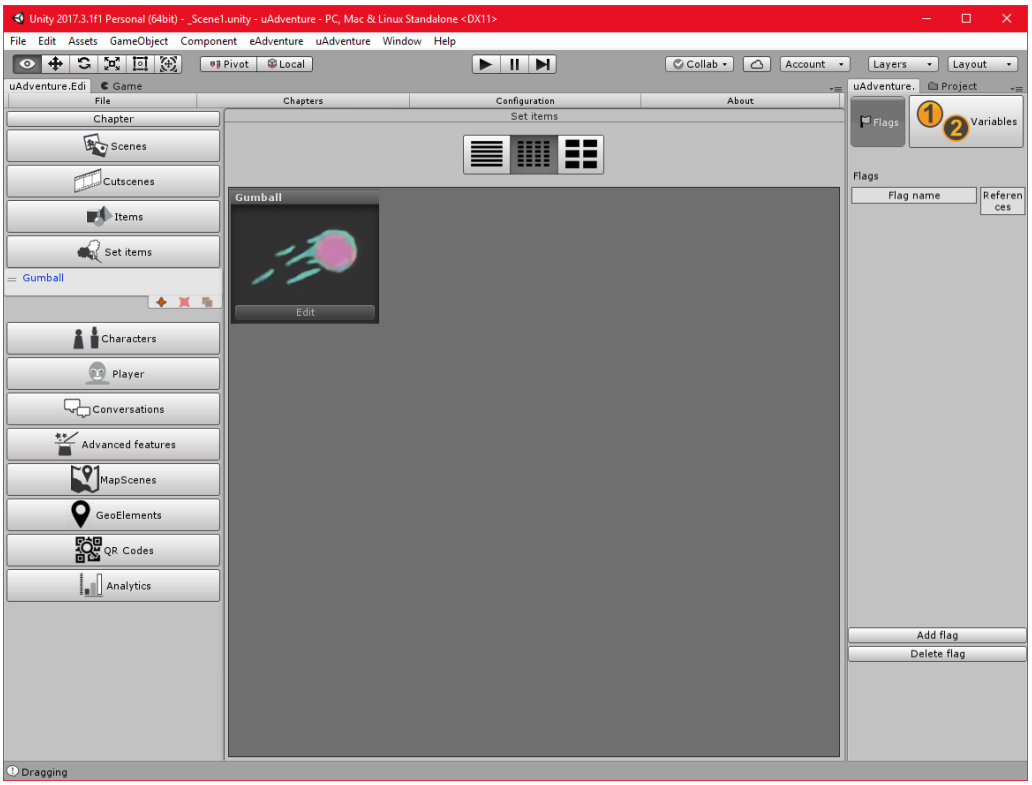

Figure 36. "Set items" general view, with just one set-item in the chapter.

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 37). There are no restrictions to the size the image of the set-item must have.

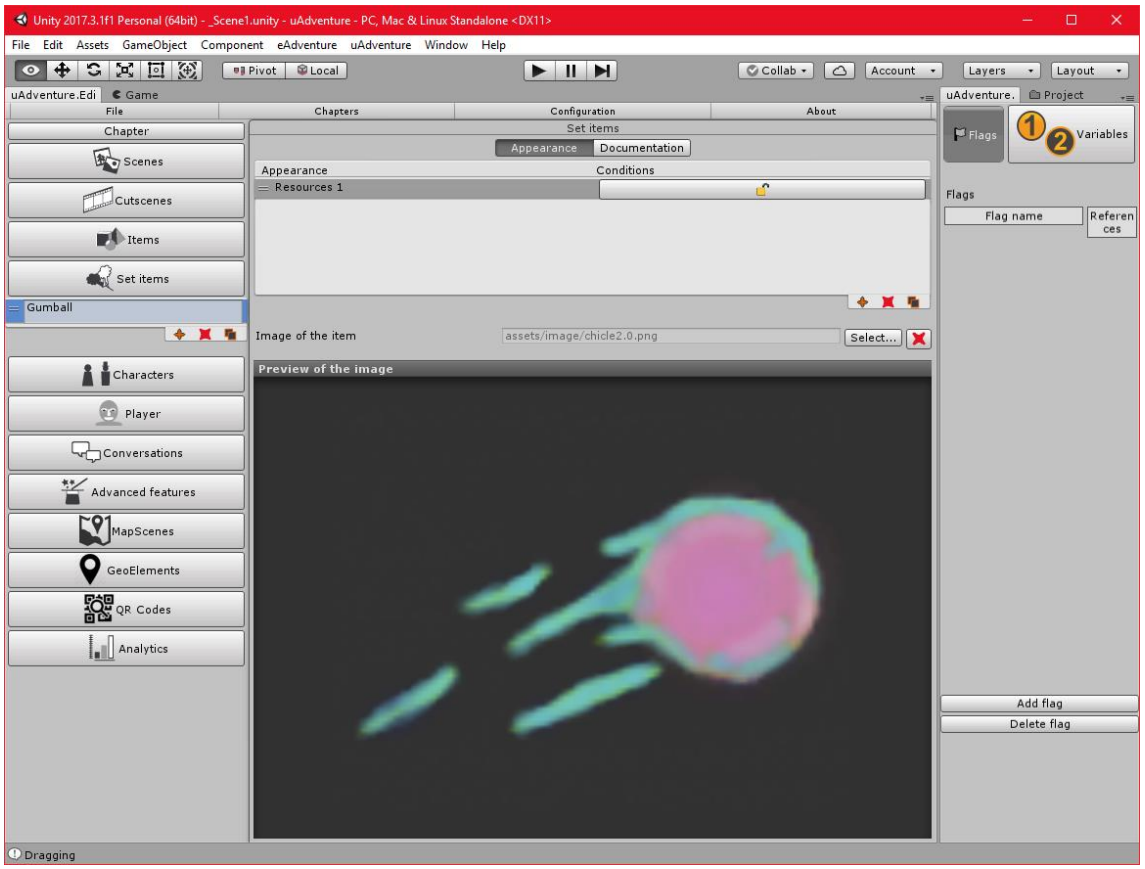

Figure 37. Set-item edition panel in the "Appearance" tab after setting the image.

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 2.4.5), 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. 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.

2.6.1. 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 much more resources. This means that different animations must be provided for different situations (talking, grabbing, and walking) and with different orientations (Figure 38).

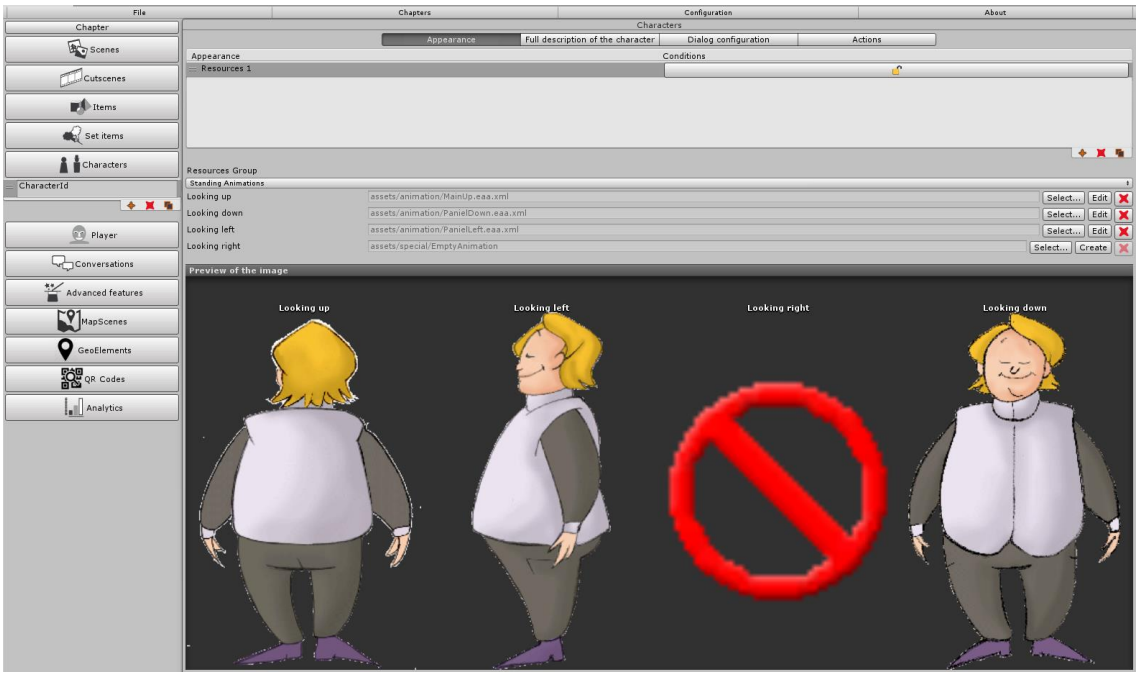

Figure 38. Character edition panel in the "Appearance" tab allows for the edition of the characters animations.

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 doesn't have a visual representation for its left view, a vertically

mirrored version of the right animation is used instead. Same happens when the right view is missing but the left view is present.

2.6.2. 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 39):

- Show speech bubble: If selected, the dialogs 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.

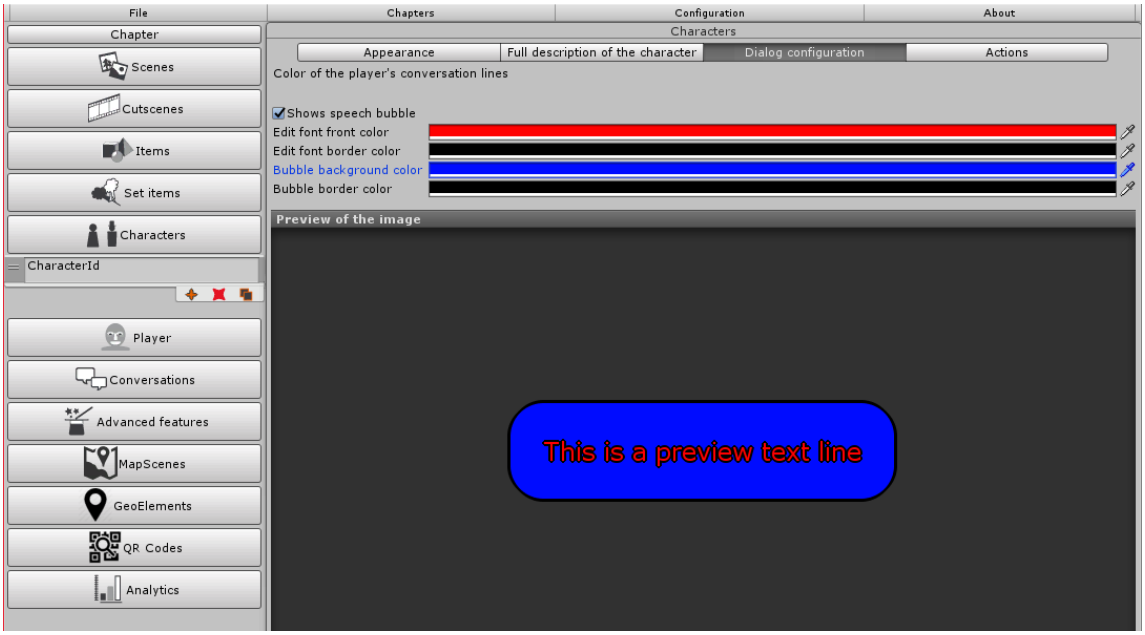

Figure 39. Preview of the text line of a character (or the player).

2.6.3. 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 2.4.5).

2.6.4. 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. Also, even the talking animation is going 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 40.

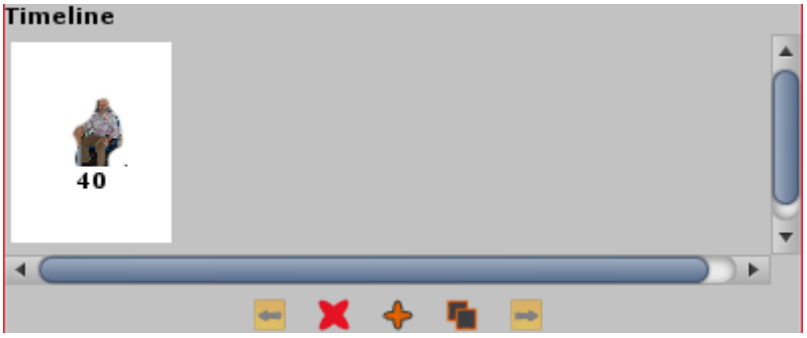

Figure 40. Balta standing animation.

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 41.

Figure 41. 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 42.

Figure 42. Resources for the different scenes.

Finally, in the next Figure 43 we can see the resulting scene exit flow.

Figure 43. 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 2.4.5, adding a NPC reference this time to Balta character.

2.7. Conversations

Just like the player interacts with items, the player might also interact with characters. However, the kinds of interactions are significantly different. In the "Actions" tab of the characters we find an 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 are created for: guiding the player, providing information, or evaluating the player.

More than one character, besides the player, 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 do this, first a new conversation element must be 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 and player in the given order. The node will show all the lines inside of the node (Figure 44).

Figure 44. 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 45).

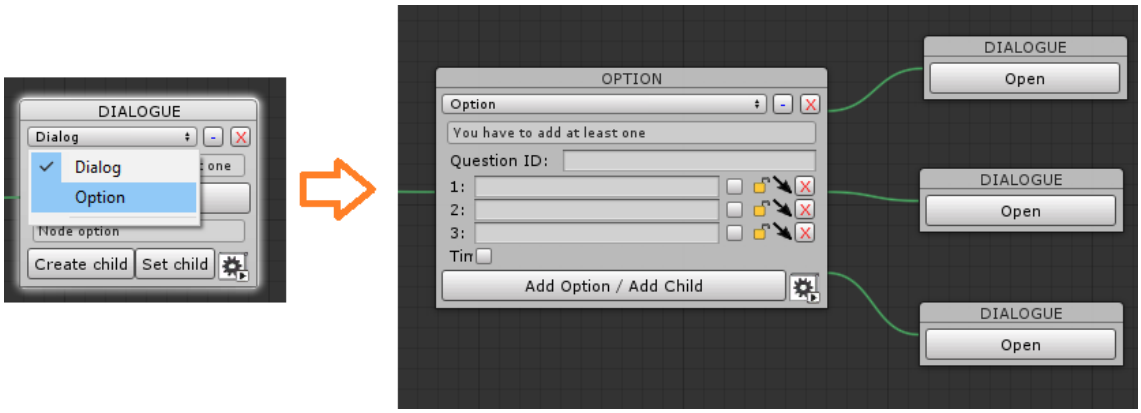

Figure 45. Options node with three children.

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 45).
- Collapse button: Any node in a conversation can be collapsed by clicking on the "-" symbol on the top right of it (Figure 46). When a node is opened you can see the different contents it holds depending on its type.

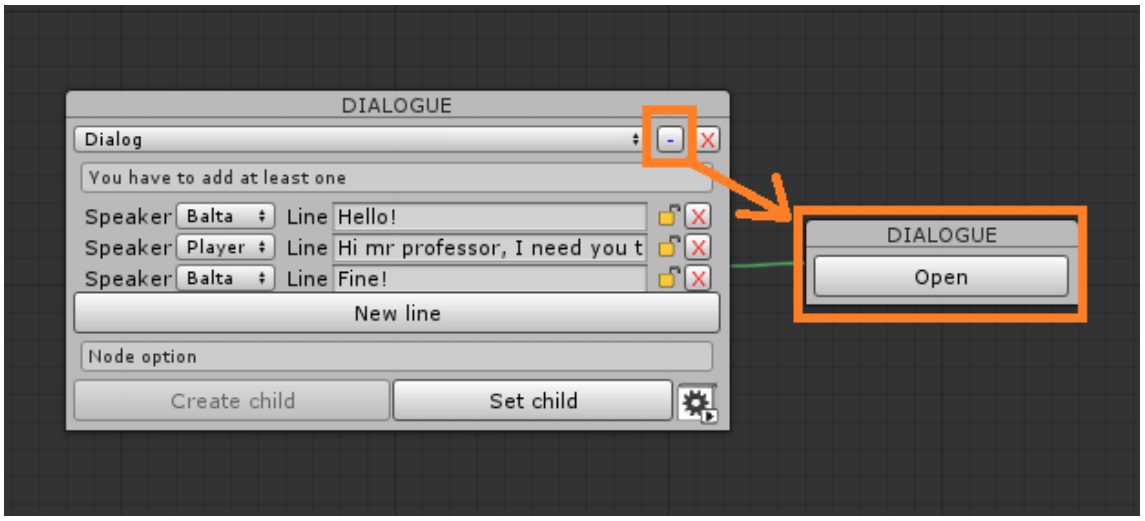

Figure 46. Collapsing node process.

- Delete button: By clicking on the "X" symbol, it will delete this node and all its children.
- 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 ( ).

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 47). 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 48).

Figure 47. Selected node dialog box.

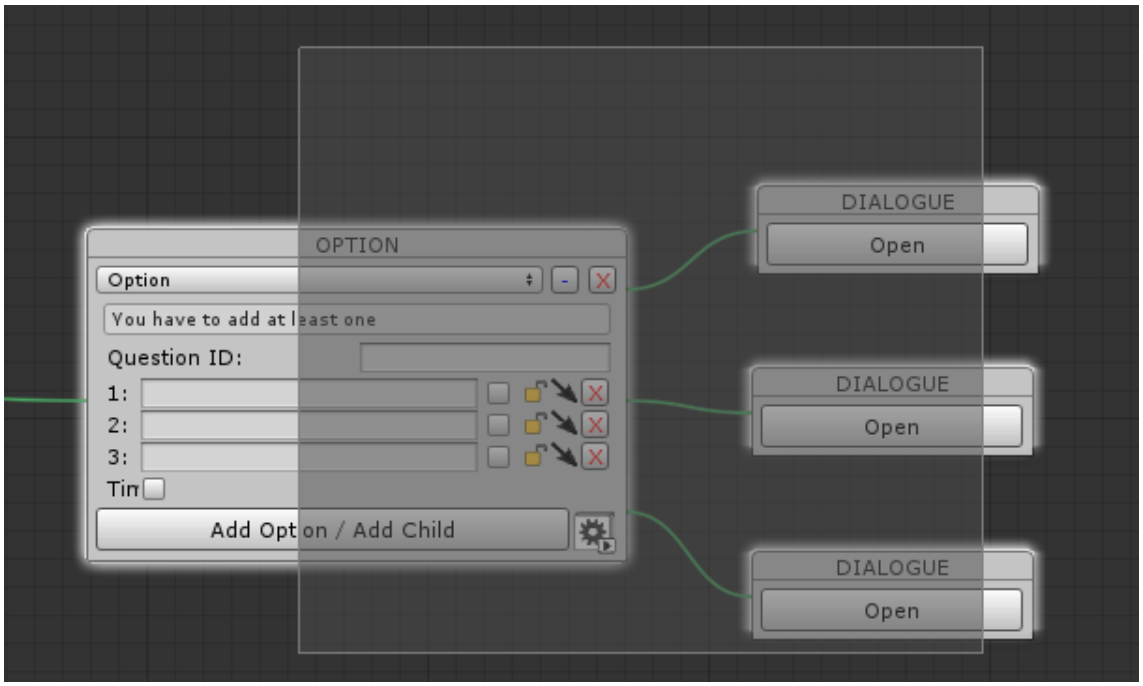

Figure 48. Multiple node selection tool. When multiple nodes are selected, the drag is applied to all of them at the same time.

2.7.1. 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.
- Line: The text to be talked by the character.
- 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 doesn't 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.

In contrast to <e-Adventure> all the TTS and sound options have been disabled for now but will be possible implemented soon.

2.7.2. Options node contents.

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

- "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 lines between nodes will change its color depending on the status: blue for the new link and red for the link that is going to disappear (Figure 49).

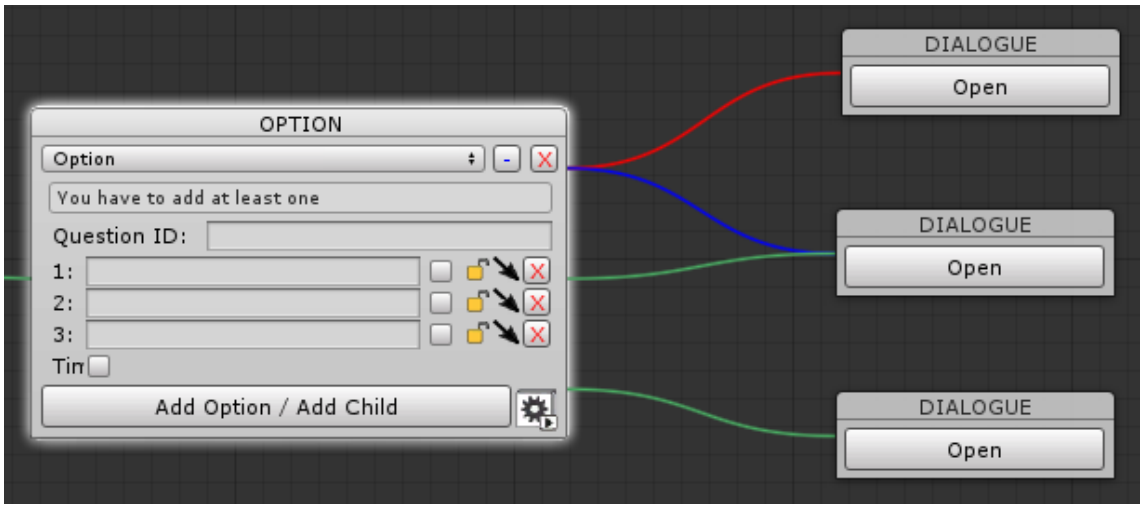

Figure 49. Child node selection: blue means new link; red means that the link is going to be deleted.

- 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 can also have a child node as any of the other options, that will be followed when the time runs out.

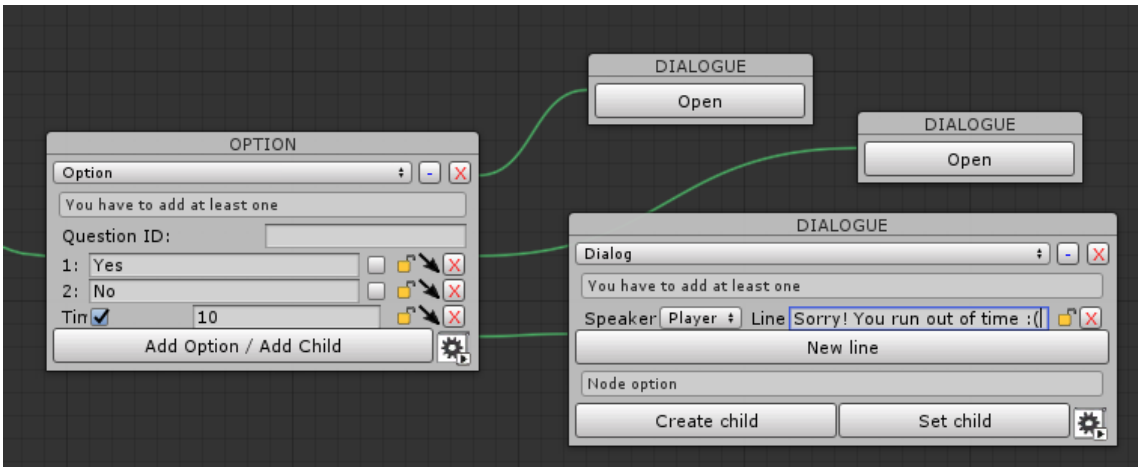

Figure 50. Timer option enabled.

Note that, for options, new options are available in contrast to <e-Adventure>. In <u-Adventure>, we want to make as easy as possible the assessment process. Because of this, it is very easy to add assessment for analytics in each question. In order to add some tracking information to an options node (i.e. because it 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 questions are the correct ones.

2.7.3. 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 51):

- 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?

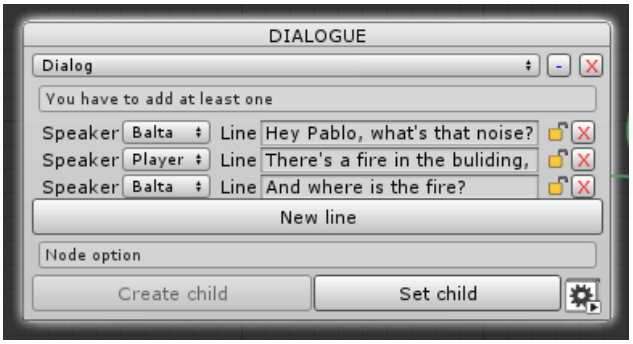

Figure 51. First conversation.

Now we are going to add three different options (Figure 52):

- Right way: "In office 411".

- Panic way: “In office 411. It’s a huge fire, every man for himself!”. (In this option we will just go for a single dialog line for Balta saying “Help! Run!”. This dialog node should change flag values to set the game state to “Panic” but this will be later explained in 3.1).
- No progress: “That doesn’t matter at all”. (In this option, we are going to configure the node to go to a dialog node and make Balta say: “You’re not helping me. I can’t know where to go!” and go back to the options group by using the “set child” button and selecting back the first options group.)

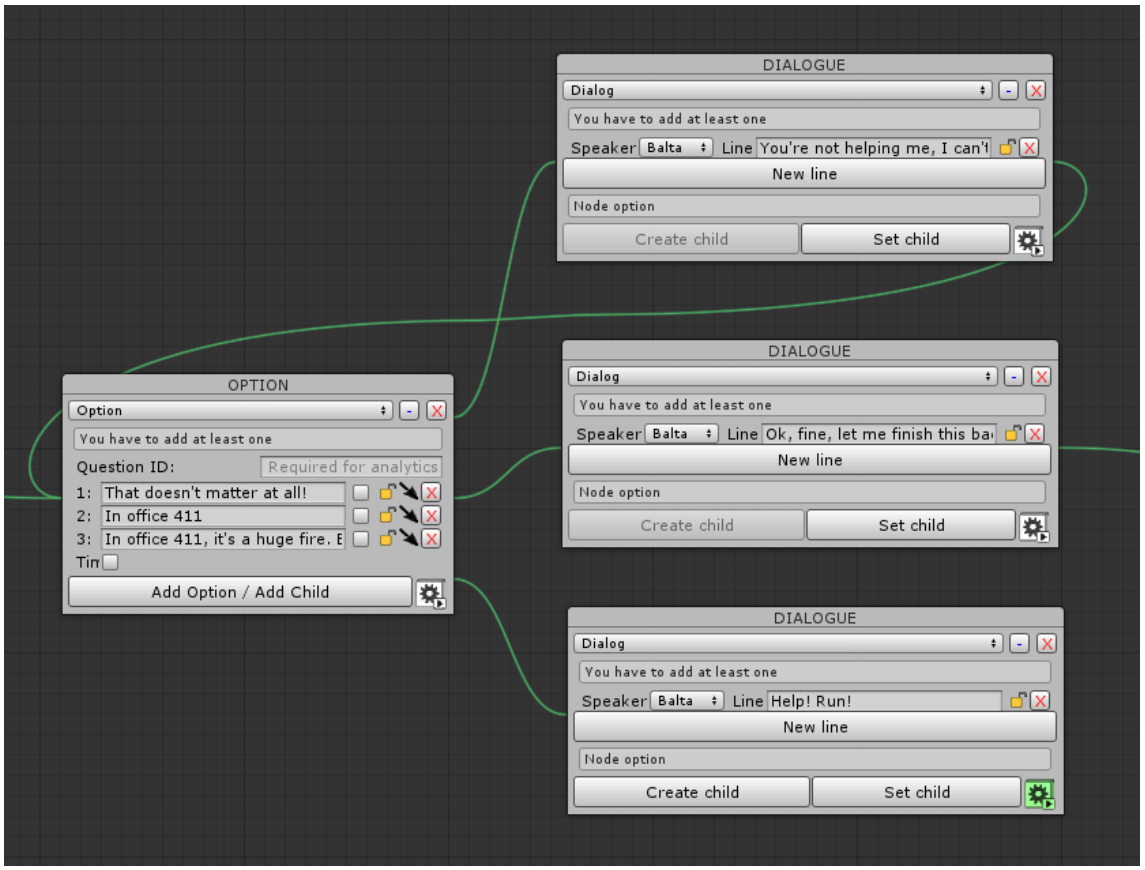

Figure 52. First option set. The top dialog is connected back to the option node, the second continues forward and the third ends the conversation there.

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 executed always, no matter what choice is used (Figure 53).

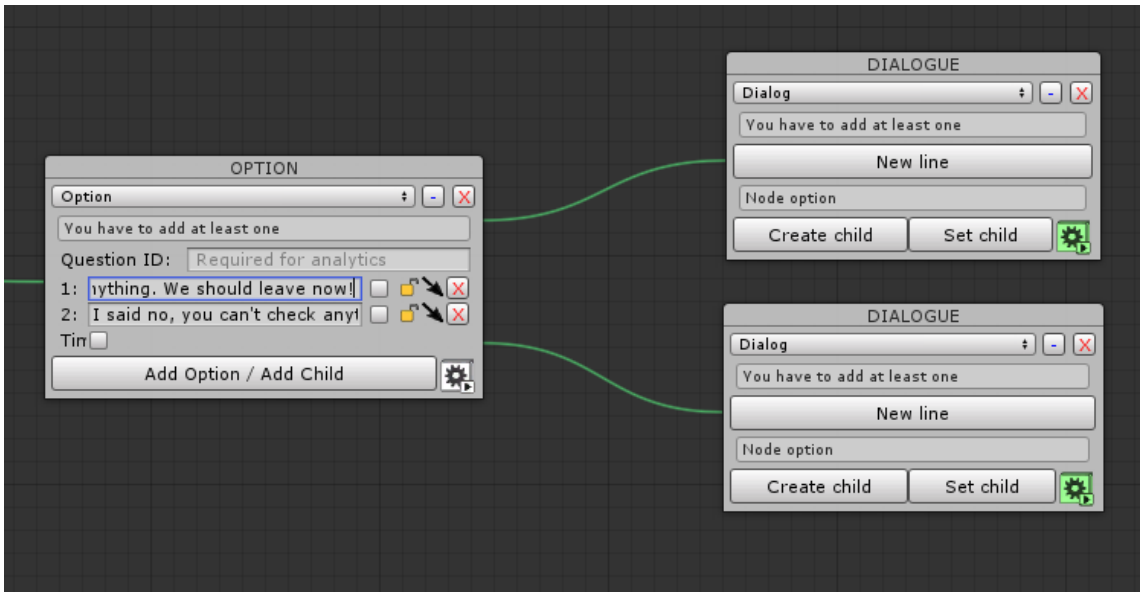

Figure 53. Last option group. The dialogs for each option are left empty, but the effects are used to change the game state.

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 54). Select it and press "OK".

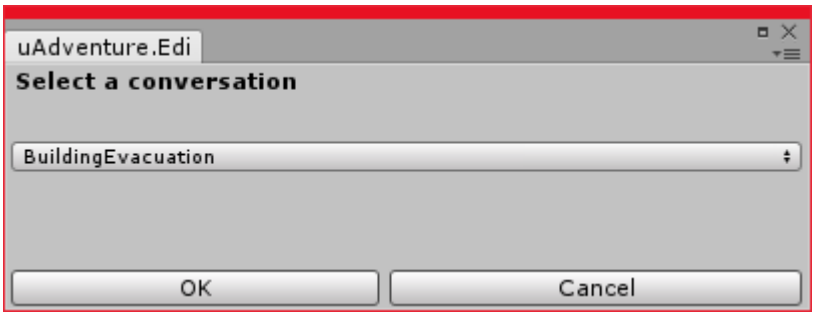

Figure 54. Select Conversation prompt.

The new action will appear first in the list (Figure 55).

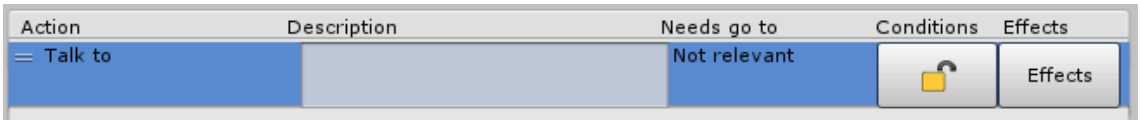

Figure 55. Talk to action added to Balta character.

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. If you want to know more about effect before continuing, check 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 56).

Figure 56. 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.

2.8. The player

The player, in third person games, is the on-screen avatar of the real-life player. In first person games, only a few of the configuration values are available for the player. This is a character that responds to player input and interacts with the different game elements. This player character can be configured just like any other character, except for the fact that it lacks actions.

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, they are not enough by themselves to create an interesting game. Games require a narrative, a story to be told. In <e- Adventure> the story flow is managed by *flags* and *variables*.

3.1.1. Flags

Flags work as switches, at any point in the game can be either **active or inactive**. 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. On the other hand, variables can have integer values and can also be used as part of conditions, depending on whether they are greater, smaller or equal to a given value.

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**. By default, variables start the game being 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 does not have an implication in the game. Conditions are introduced to give a purpose to a variable or a flag: a condition could be, 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 essentially two features. First, adds condition blocks which 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 57), as it can only be either active or inactive.

Figure 57. Flag condition line set for active.

On the other hand, for variables it is a little bit different, as variables have a bigger range of values, so 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.) with which to compare (Figure 58).

Figure 58. 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" were added. 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 59.

Figure 59. 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

Another, even 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 (i.e. the game over condition can be a global state Figure 60). 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" element in the structure panel to the left of the editor. "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, but using the relevant button.

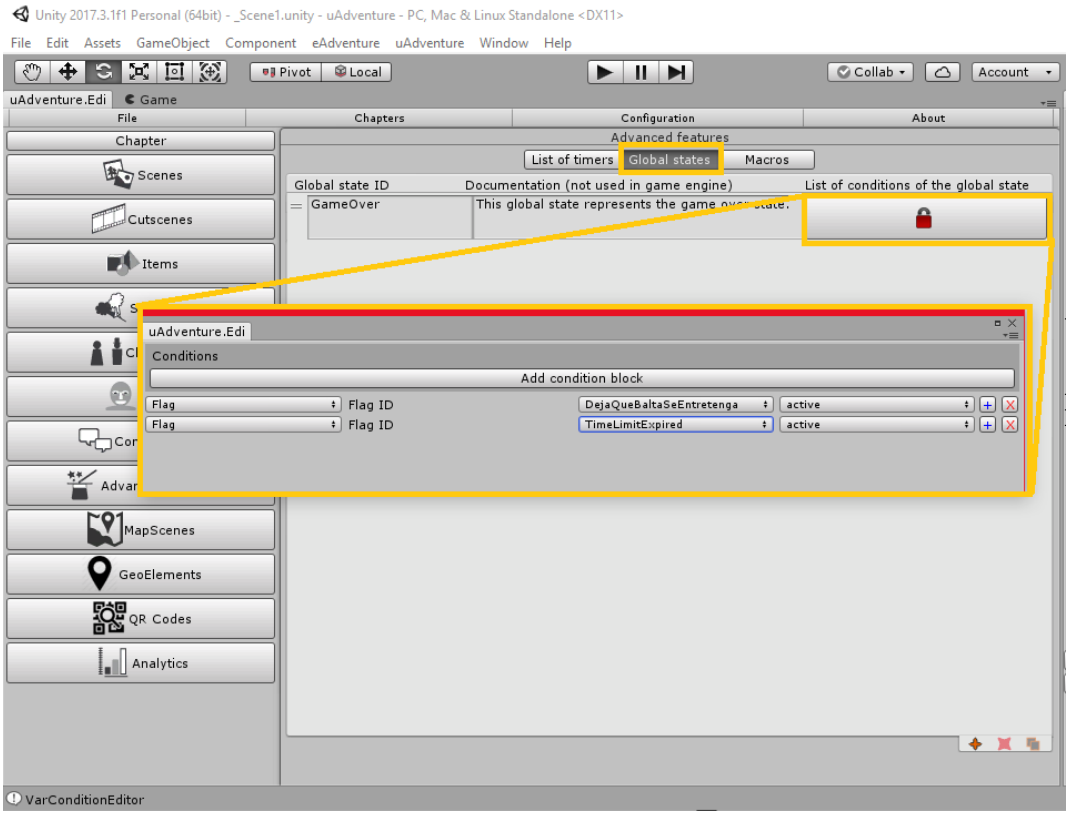

Figure 60. Global state for a game over condition.

3.1.5. What can be done with conditions?

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

- In actions (for objects and active areas)
- In conversations (for each line)
- In barriers (to activate or deactivate them)
- To change the behavior of an exit (for example, activating one exit or another in the same place)
- In references to elements, allowing the elements to appear and disappear as needed
- To change the appearance of an element, 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 and a use action. The idea is to identify that the phone has been answered so, when used, it changes its appearance to the not ringing one. To do it, we are going to use a flag that we are going to call “PhoneAnswered”. This flag, when active, will indicate that we already answered the phone so it can change back its appearance to normal. Also, in the next example we will use this flag too to block the exit of the office.

First, we must go to the “Flags and Variables Window”. If you are using the uAdventure layout, this window is already docked 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 61).

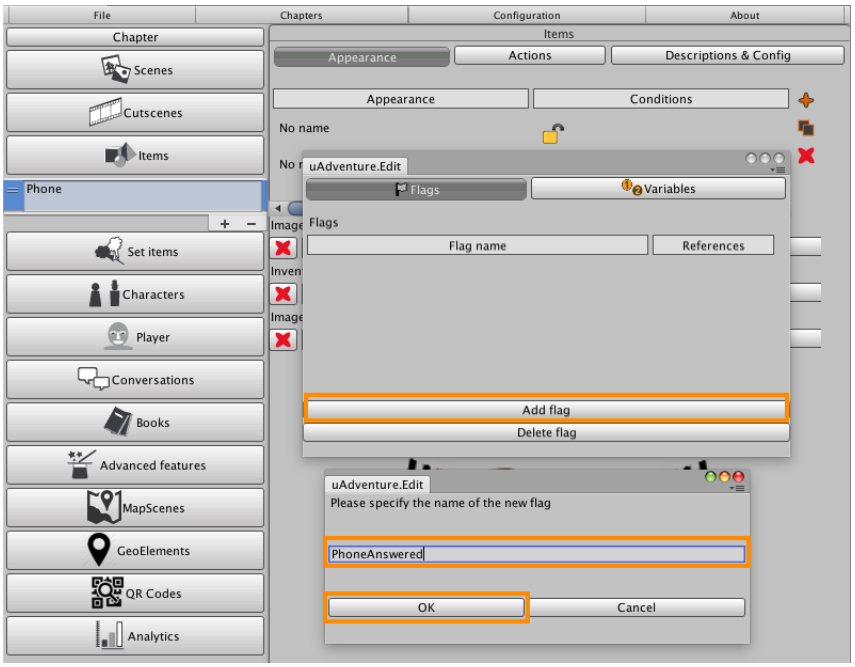

Figure 61. Flag "PhoneAnswered" creation process.

Now we are going to use this flag in the resources of the phone to change them to normal state. 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 condition is not true, it will not be used anymore, leading its usage to the second resources.

We can put it in practice by going to the “Phone” item in the structure and switch to the “Appearance” tab. As earlier, 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 62.

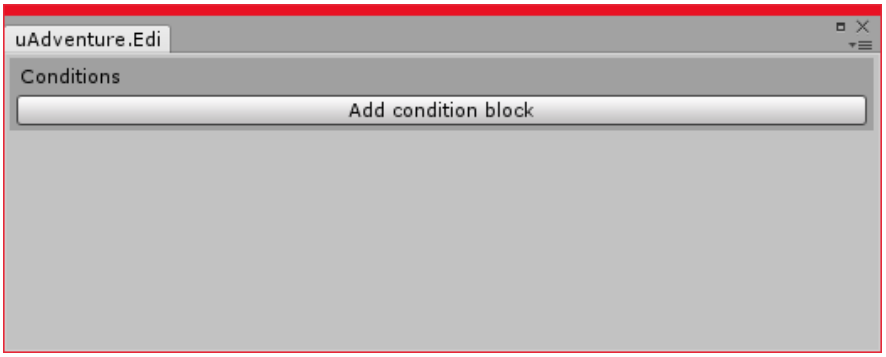

Figure 62. Conditions window.

We can explain a little bit more about the condition window. Conditions are made by blocks. A block can be made true by any of its former conditions individually. However, for a condition to be true, all its blocks must be true. Therefore, adding blocks to the condition is the same thansaying “and” when it comes to conditions while, adding conditions to a block is the same than saying “or” in each block.

Go ahead and add your firs block by clicking “Add condition block”. You will see a line appear right below. 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 63). This way, when the flag becomes active, the phone will change its appearance to normal.

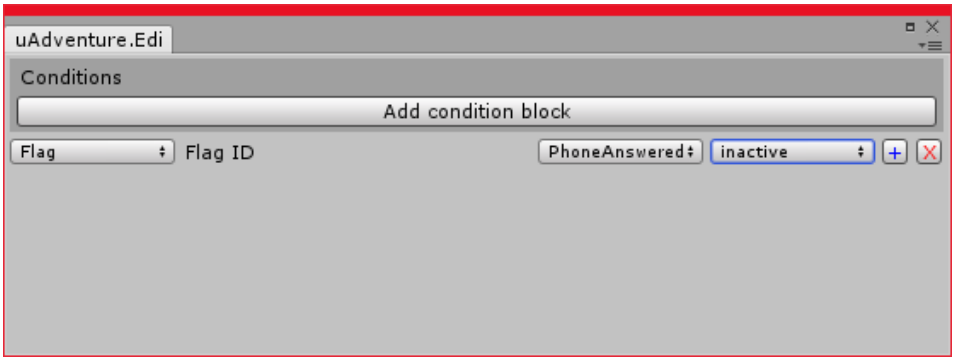

Figure 63. Conditions filled with "PhoneAnswered" flag inactive.

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 in the conditions field to edit them. Inside, let’s make the condition match “PhoneAnswered” is active as in Figure 64.

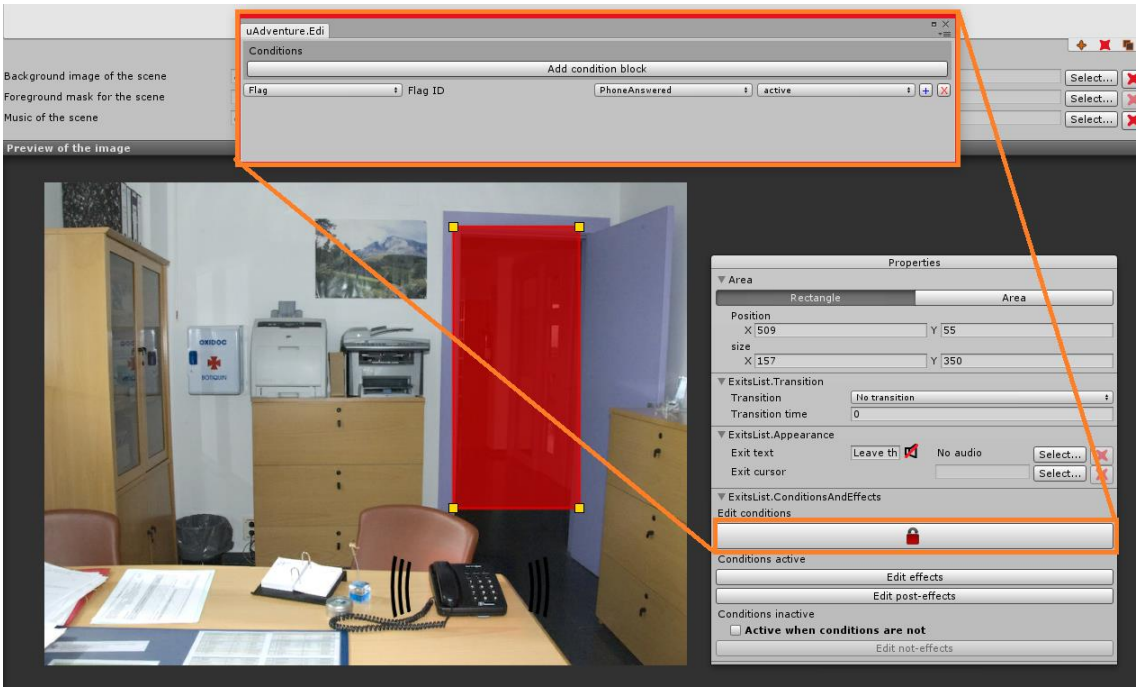

Figure 64. Editing the exit conditions in the element inspector.

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” however, are used only after we arrive into the destination scene. Finally, and this is the most interesting for this example, are the “Not-Effects” that are executed when the exit cannot be passed.

We are going to use “not-effects” to show a message to the player letting him know that he must

answer the phone before leaving. To do it, first check the "Active when conditions are not" checkbox and then click on "Edit not-effects" (Figure 65). Then, add a "Speak player effect" saying, "I should answer the phone".

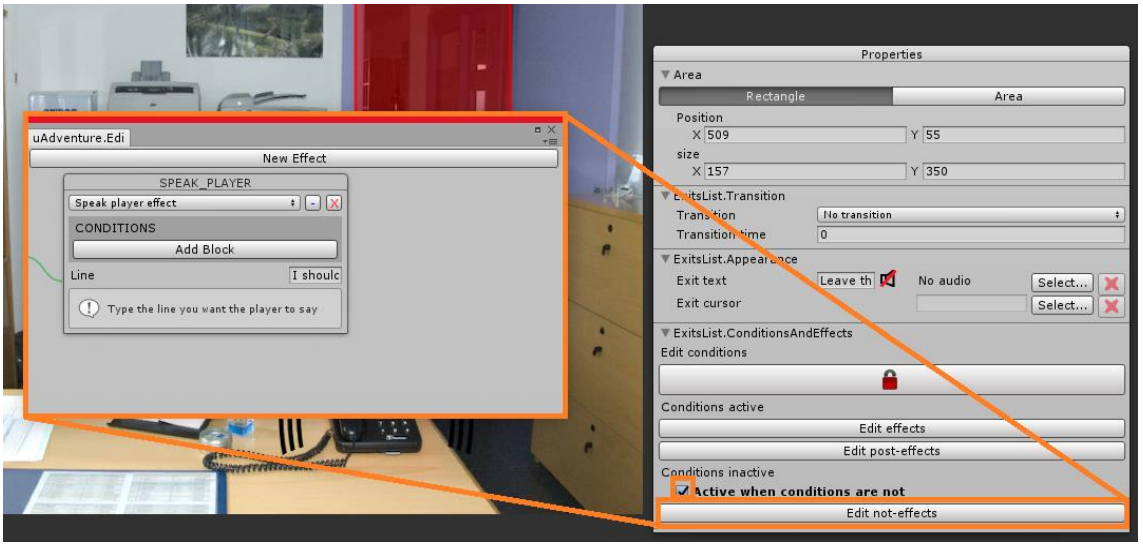

Figure 65. Speak player effect as a not-effect.

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 66.

Figure 66. 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 cutscenes 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 (Figure 67). 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.

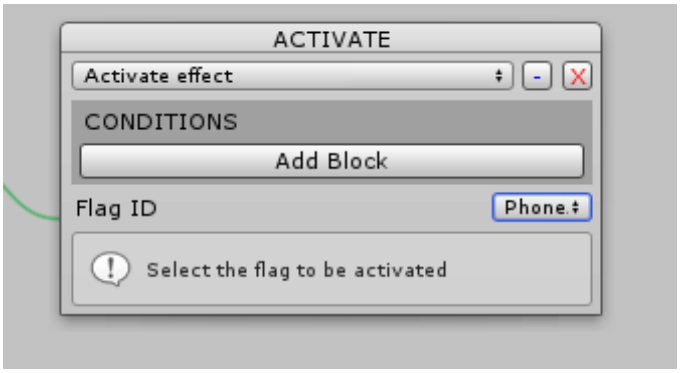

Figure 67. Effect node configured as an activate effect to activate the PhoneAnswered flag.

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 68):

- “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.

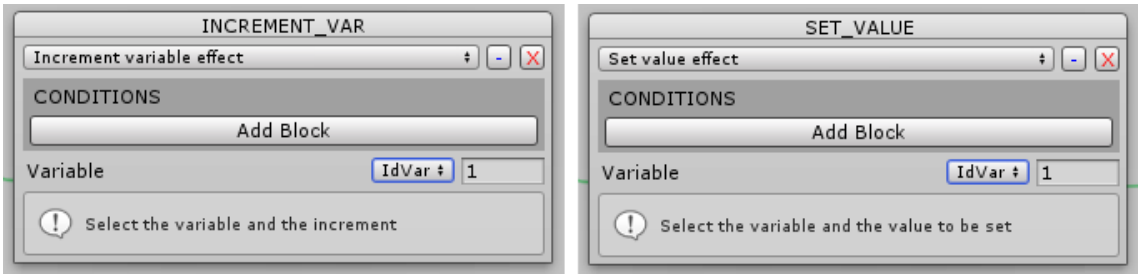

Figure 68. Effects to set the value of a variable or increase its value in 1 units.

3.1.10. Other effects

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

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	
Show text	Displays the given phrase in an specific point of the scene for an 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 (cutscenes 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 cutscene	Plays the selected cutscene.	Allows to trigger cutscenes 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 similar).
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).	
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.	
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 a list of two. 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	

Figure 69. Table with the effects available in <u-Adventure>.

3.1.11. EXAMPLE: Triggering a cutscene 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 cutscene "PhoneConversation". Now that we know how effects work, we can use a "Trigger cutscene" effect to open our cutscene.

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 cutscene". When selected, we are going to be able to select the "PhoneConversation" cutscene from the list below.

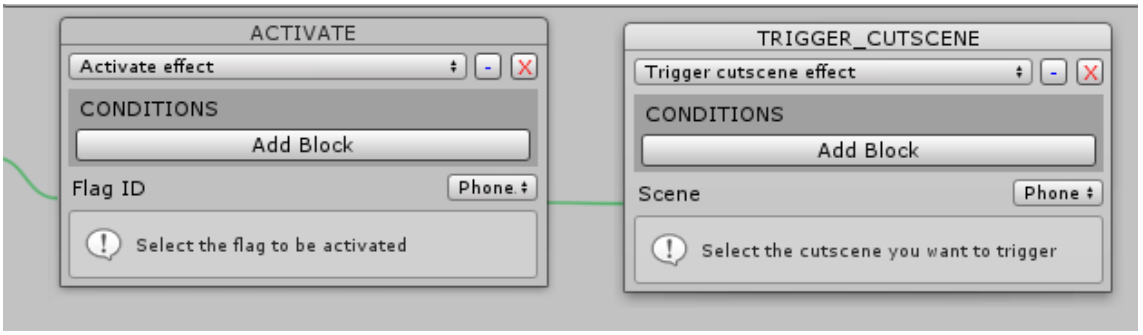

Figure 70. New Trigger cutscene effect appended after activate effect.

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 and the documentation too. 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.

3.2. Organization of the element references in the scene

Section 3.2 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 and 3.2.3 cover settings of the reference list. Section 3.2.4 addresses the preview panel.

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 71).

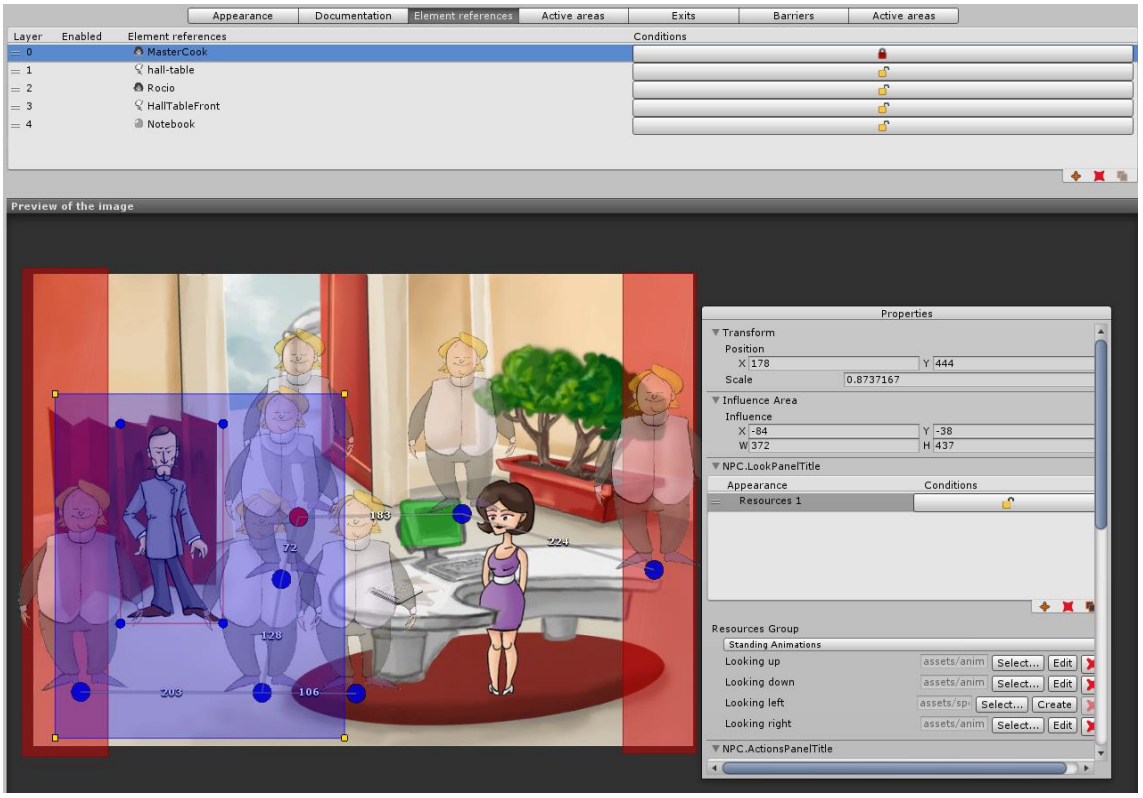

Figure 71. Edition of the layer of element references.

The layer of the player is treated in a singular manner. Since the player can move in the scene, the <e-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 to 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 get drawn before an element when it is supposed to be farther. In Figure 72 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 72. 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 down-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 popup lets the user select any of the existing elements. However, for elements to be added, first they must be created. For more information on how to create each type of element see sections 2.4.1 (items), 2.5.1 (set items) and 2.6.1 (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 appear 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 to 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 73). They are defined in the same way that exits where, and can be of different sizes and in different positions.

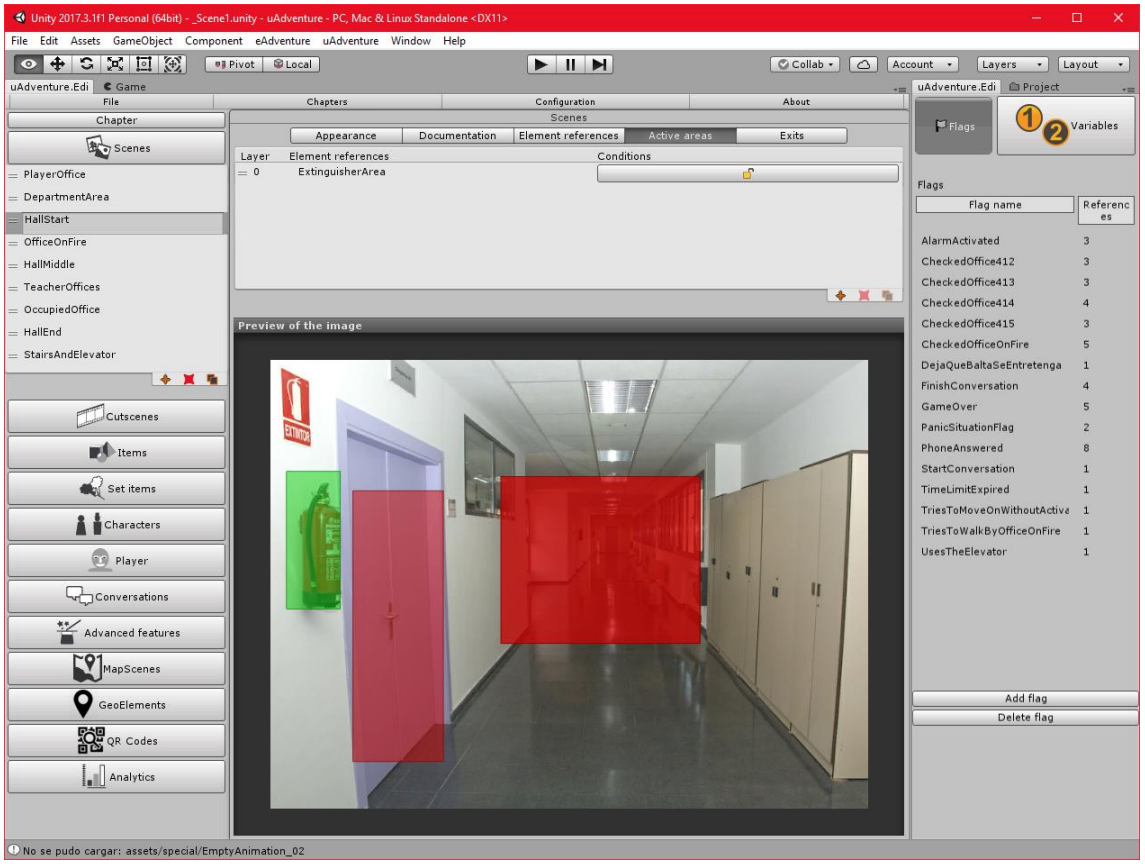

Figure 73. "Active areas" tab in the scene edition panel. Active Areas ca be identified in green while exits are red.

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 to not be accessible 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 will appear in the middle of the scene. That is our new active area, so now move it and resize it until it fits the fire alarm. The result should be as in Figure 74.

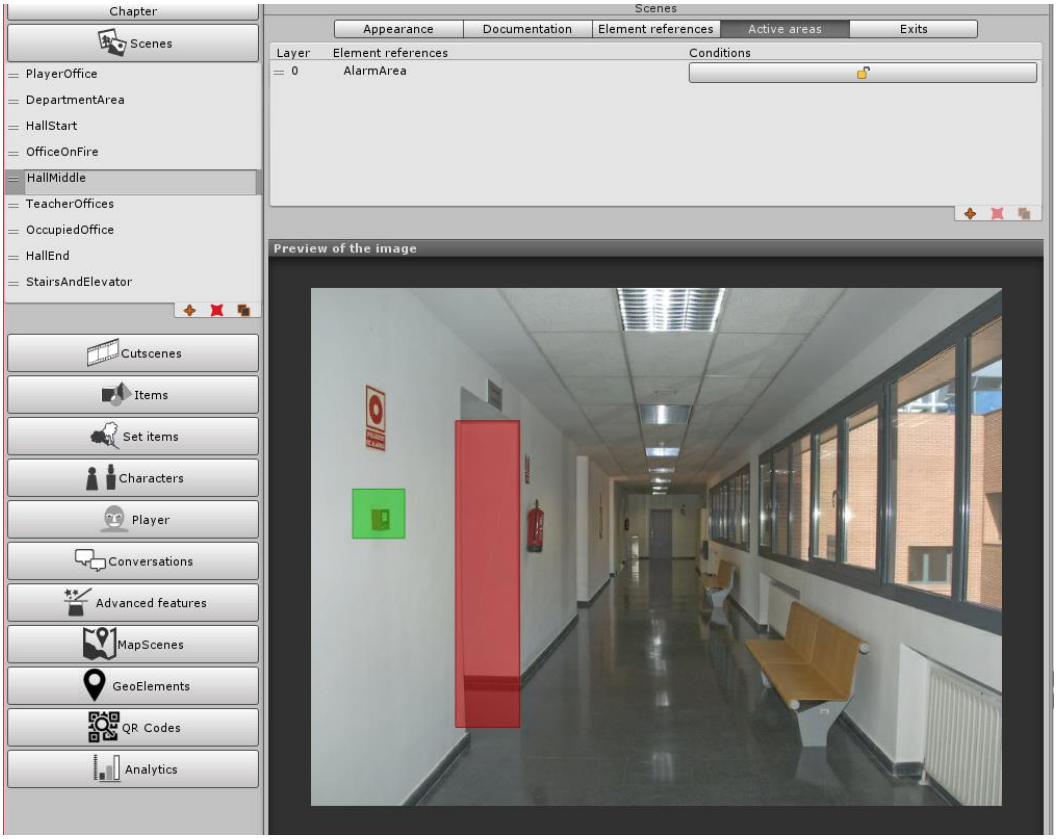

Figure 74. Fire alarm active area.

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 need 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. Inside of 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 75.

At this point, we could give conditions to the effects inside of the actions to change the text that is shown when performing the action. However, there's 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 area matched is used. Hence, by adding another action with the same type and setting up the conditions of each one, we'll be able to switch from one action to the new one. To do this, we're going to open the conditions tab on the first action and add the condition of "AlarmActivated" is inactive, it should be enough to disable it when clicked. In our second use action we're going to add just an "speak player effect" saying that the alarm has been already activated.

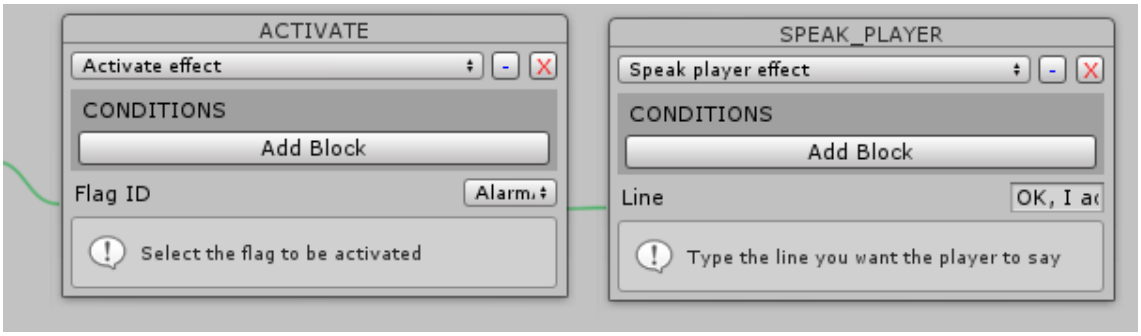

Figure 75. Activate alarm effects sequence.

Now by 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 are only available in third person games.

Barriers, just 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 76).

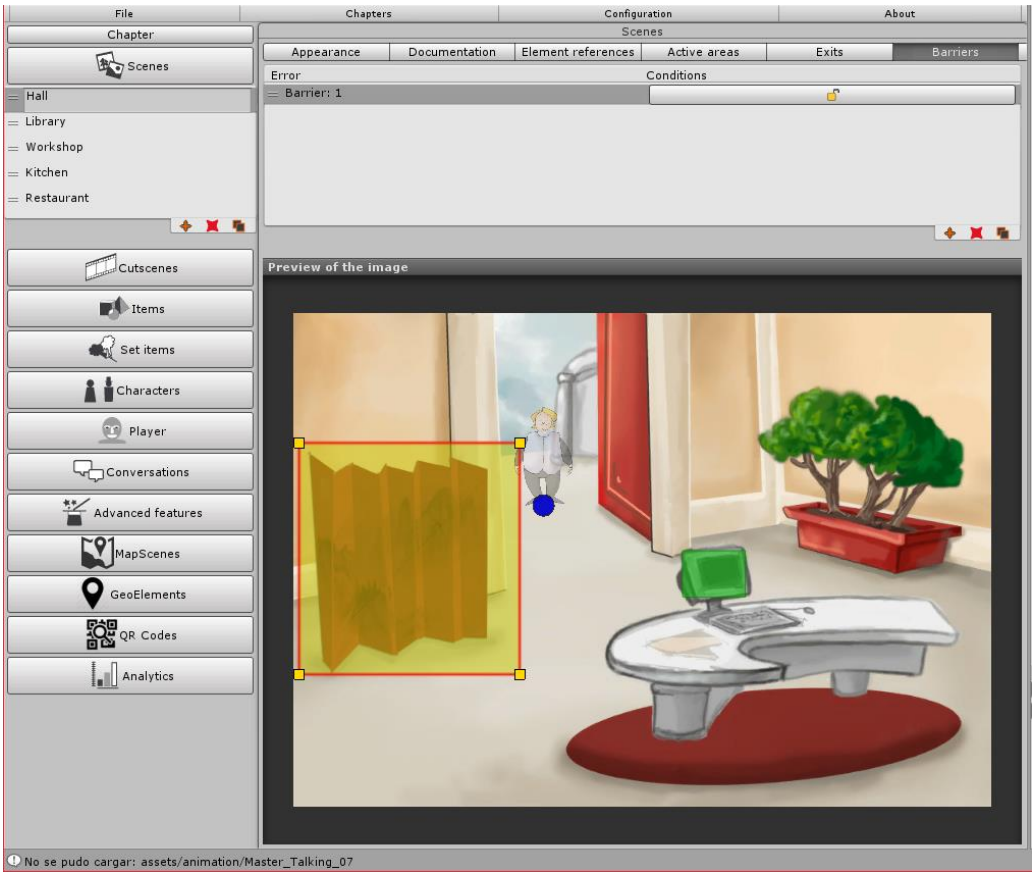

Figure 76. "Barrier" edition tab, in the scene edition panel.

3.5. Player movement

The player movement tab and trajectories are only available in third person games.

The way the player moves in a scene can be configured in two different ways: setting an initial position or using trajectories. When an initial position is used, which is the easiest way to do it, 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. This needs no further configuration and all objects can be reached when 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 77). 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.

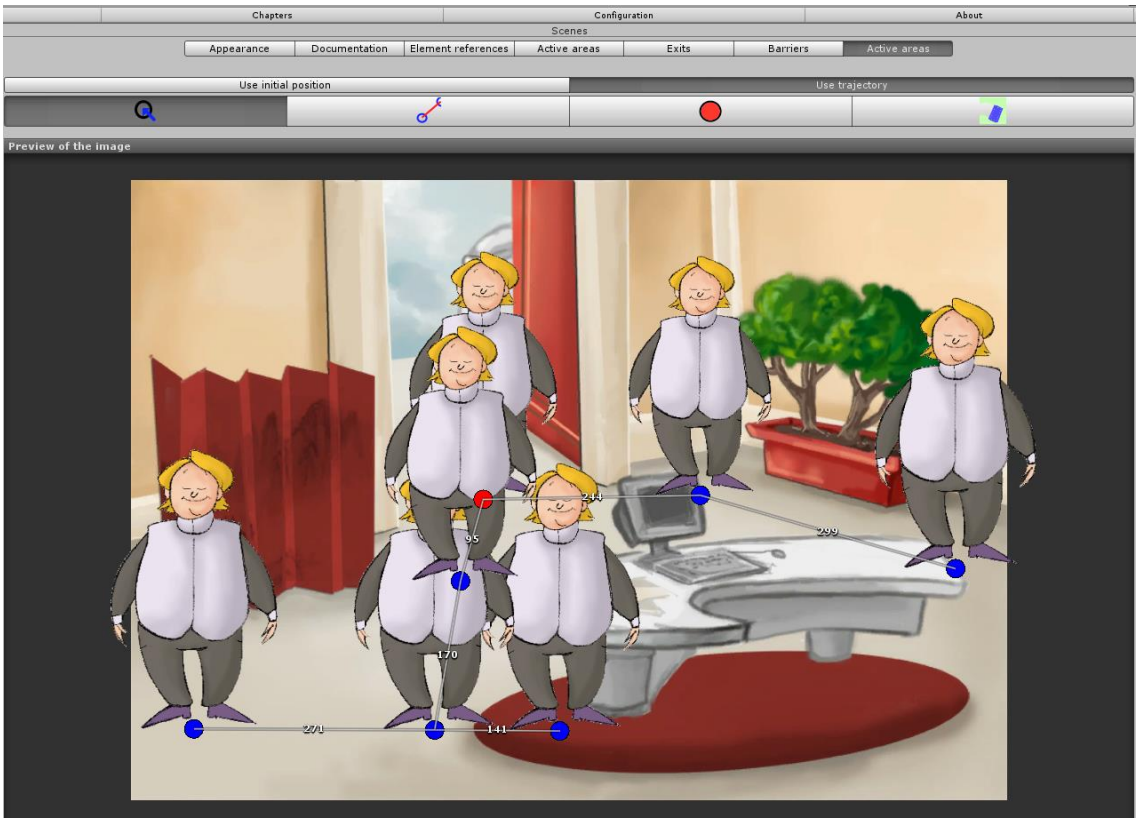

Figure 77. "Player movement" tab in the scene edition panel with trajectory mode enabled.

- 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 78). Active areas appear in blue.

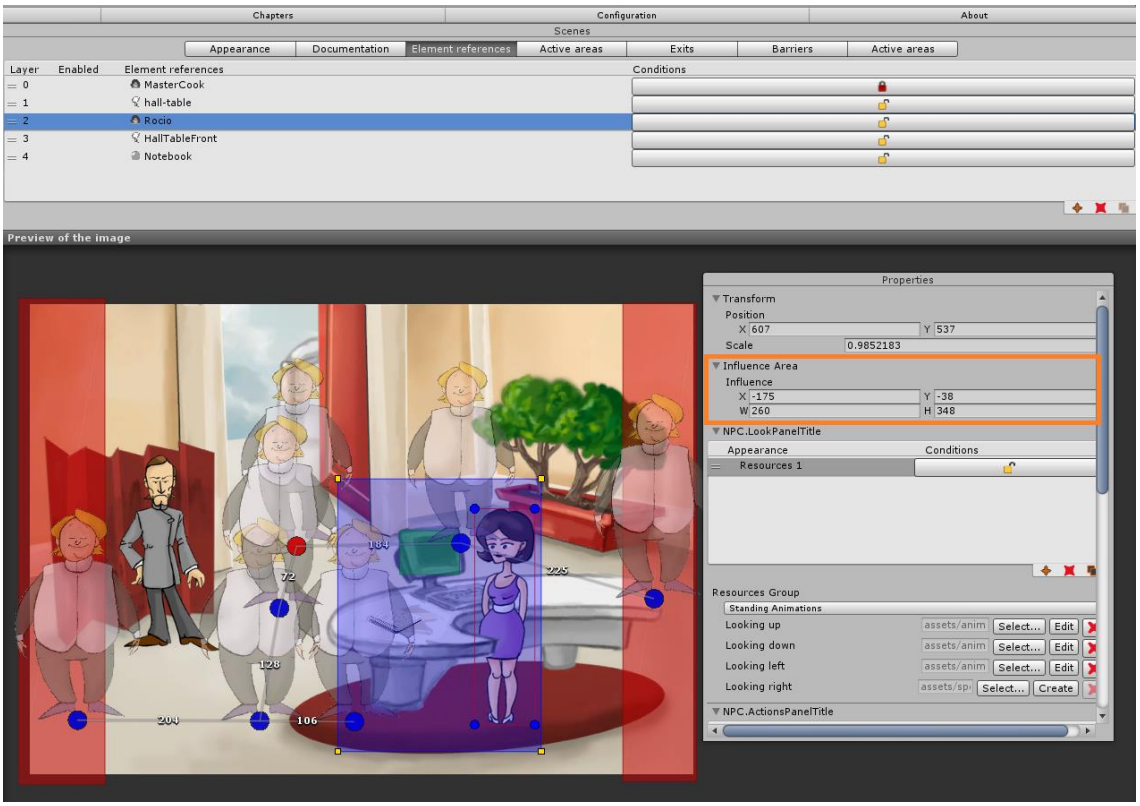

Figure 78. Editing the influence area of an element reference in the "Element references" tab.

Additionally, influence areas can be edited in the element inspector too, as its properties appear in the second position. These 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 79).

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, then 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 in a couple of ways:

- It can be defined if the timer should start multiple times or not. If this option is selected, then the timer will start counting again if the initial conditions are met at any time after first finalization.

Therefore, if a timer with this option activated expires because the time limit defined has been reached, the timer will start again if the initial conditions are satisfied. This behavior allows reusing

the timer in different parts of the game.

- The timer can be defined to run in loops. When a timer run in loops, it 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 of 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 introduced here will not be used in the games (only valid for internal documentation).
- **Loop control**:
  - Multiple starts: defines if the timer to start once or multiple times.
  - Runs in loops: defines if the timer has to start counting again after reaching the time limit or not.
- **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.

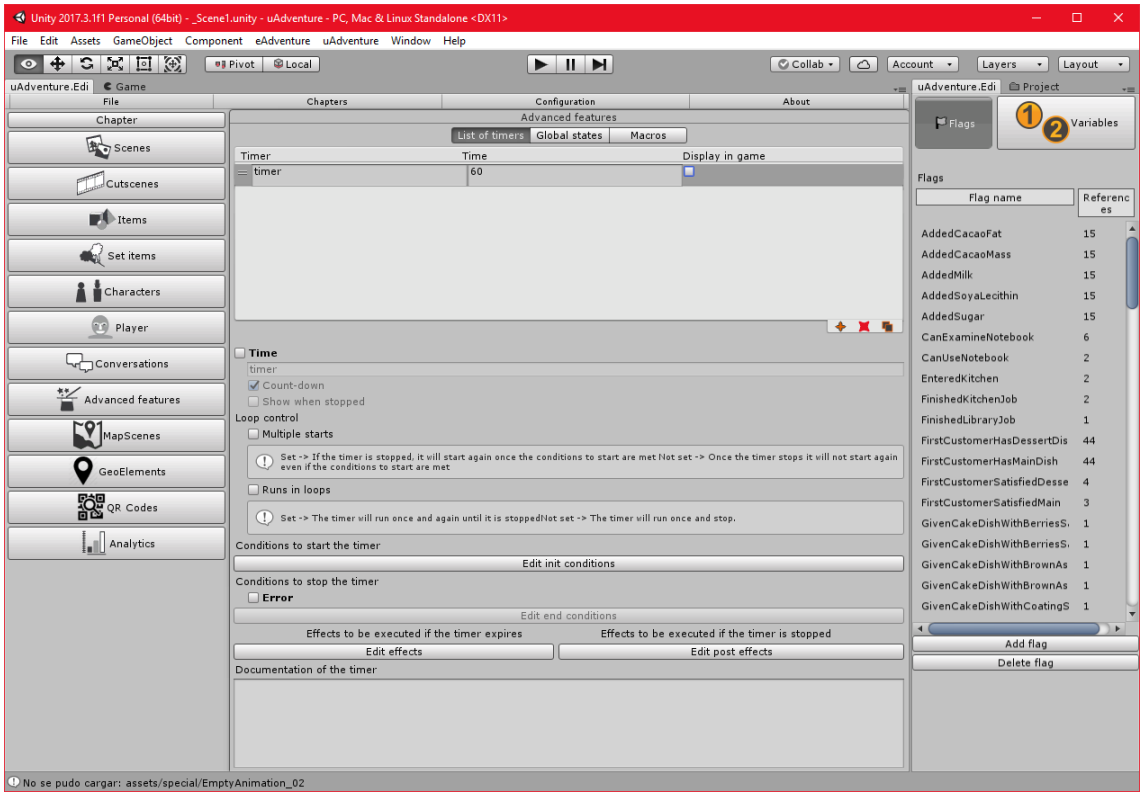

Figure 79. Edition of advanced features, "Timers" tab.

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 the own 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 elment1 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 79).

Icono normal	Icono resaltado
 btnBrush.png	 btnBrushHighlighted.png

Figure 80. 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 case it is an interaction) 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 2.3.2 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(TM)), <u-Adventure> provides support to make some slight adjustments or create simple art resources. In this section we can find any of 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 this manual in the future.

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 81).

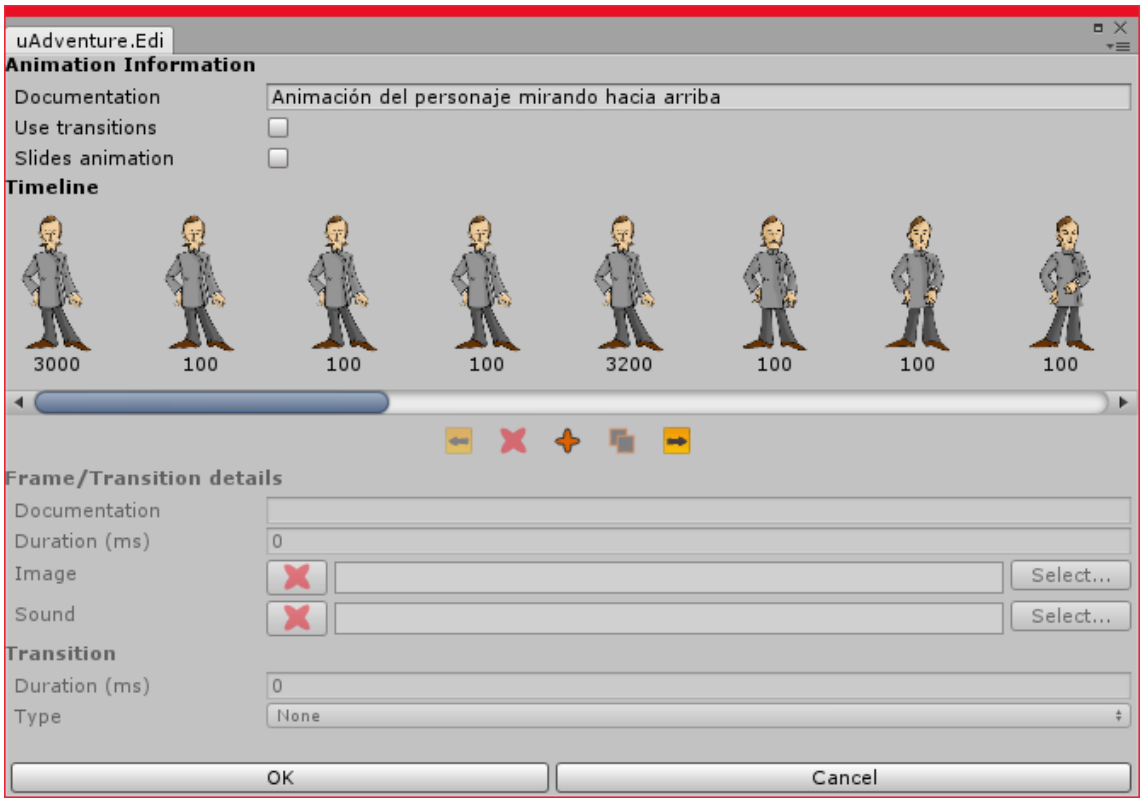

Figure 81. Animation editor.

The documentation field is used only to provide an internal description of the animation, but is not used in the game. Under 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 82):

- Duration (ms): Time that the frame will be displayed on the screen (in miliseconds). By default this value is set to 40.
- Image: The image of the frame. Any png, jpeg or bmp image is accepted.
- Sound: A mp3 or midi sound that will be played synchronized with the frame. In this manner 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.

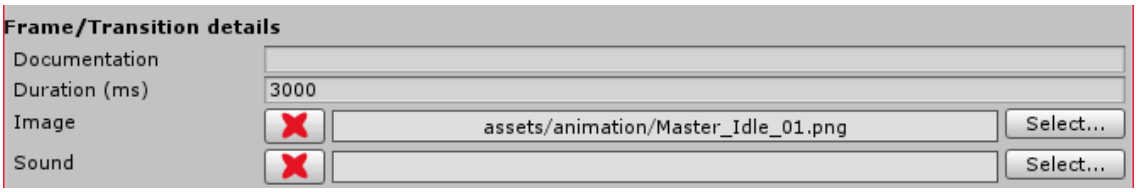

Figure 82. Properties of a frame in the animations editor.

When a transition is selected, the next properties can be edited (Figure 83):

- 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.

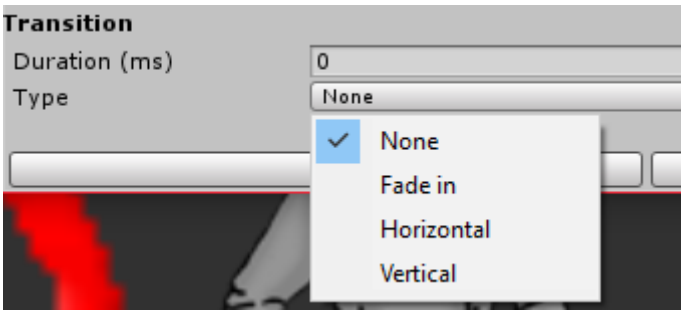

Figure 83. Properties of a transition in the animations editor.

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 84). 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 85). Finally, the third button, remove, can be used to remove a vertex.

In addition, when using the area mode, all the vertex handlers will look now light blue, and will be circles instead of squares (Figure 85).

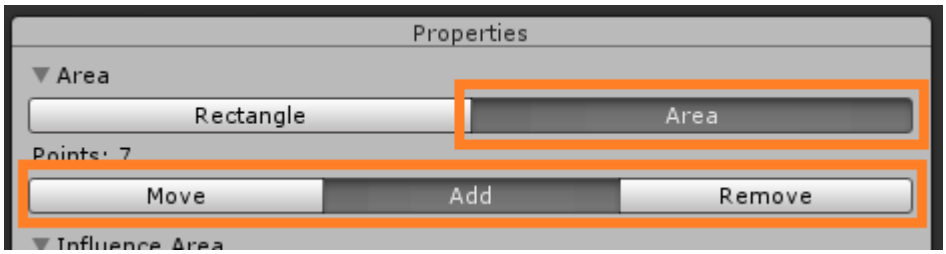

Figure 84. Area section in the element inspector. Area mode is selected. The tools are in the bottom: Move, Add (selected) and Remove.

Figure 85. 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 thereforeit cannot 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 86 to indicate the errors to the developers. If you want to share us the errors you find

you can either use the <u-Adventure> forums or go to our GitHub issues[11] 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.

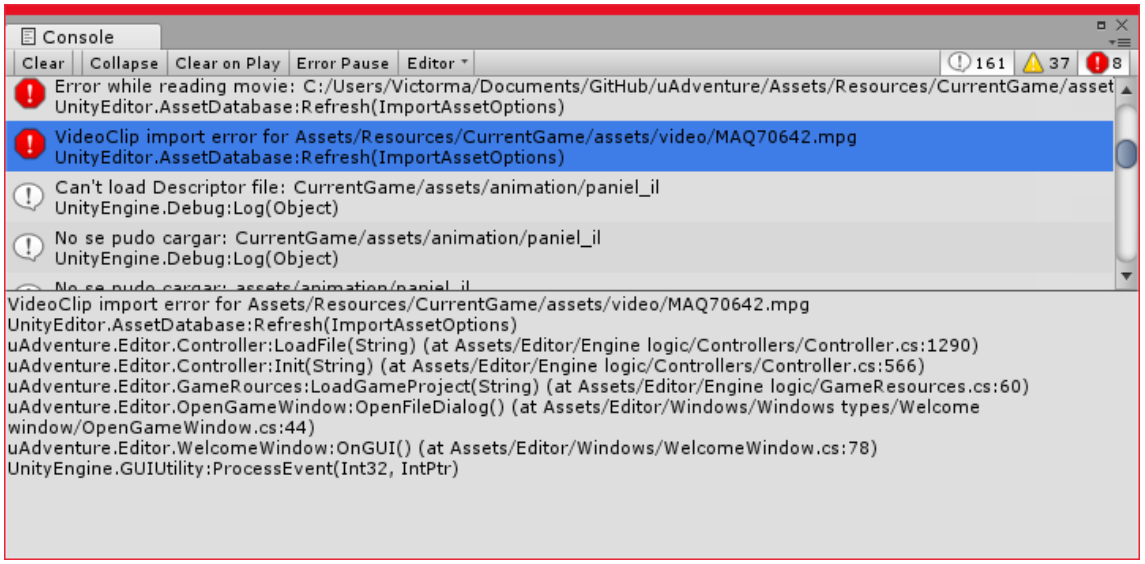

Figure 86. Unity console with an error selected.

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!

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

4. 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.

4.1. 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 1.

4.2. Adventure menu

This menu allows editing different aspects and provides general information of the game.

4.2.1. Visualization sub-menu: customize GUI elements

<u-Adventure> allows customizing some of the elements of the user interface in the games: cursors, action buttons and the inventory.

4.2.2. 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 88).

This panel is, however optional, and its ussage can be configured depending on three options in the Build window (Figure 87):

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

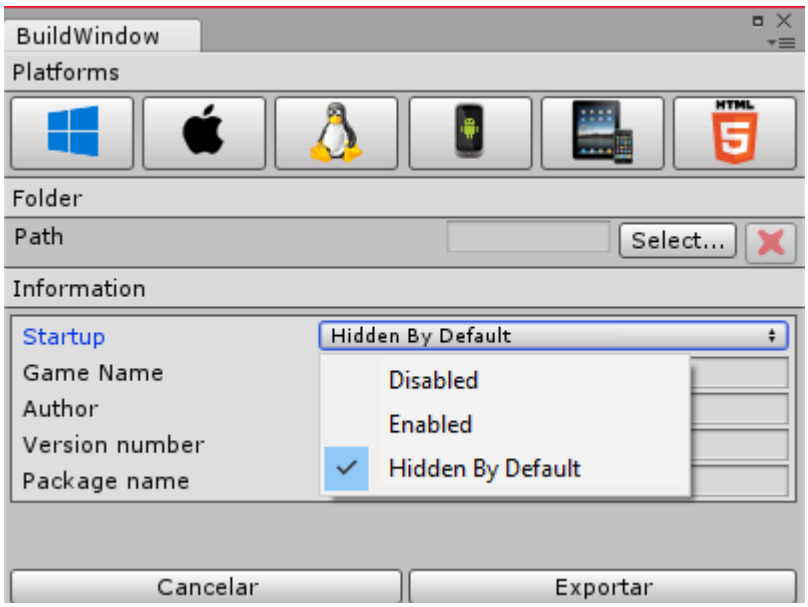

Figure 87. Build window, startup configuration.

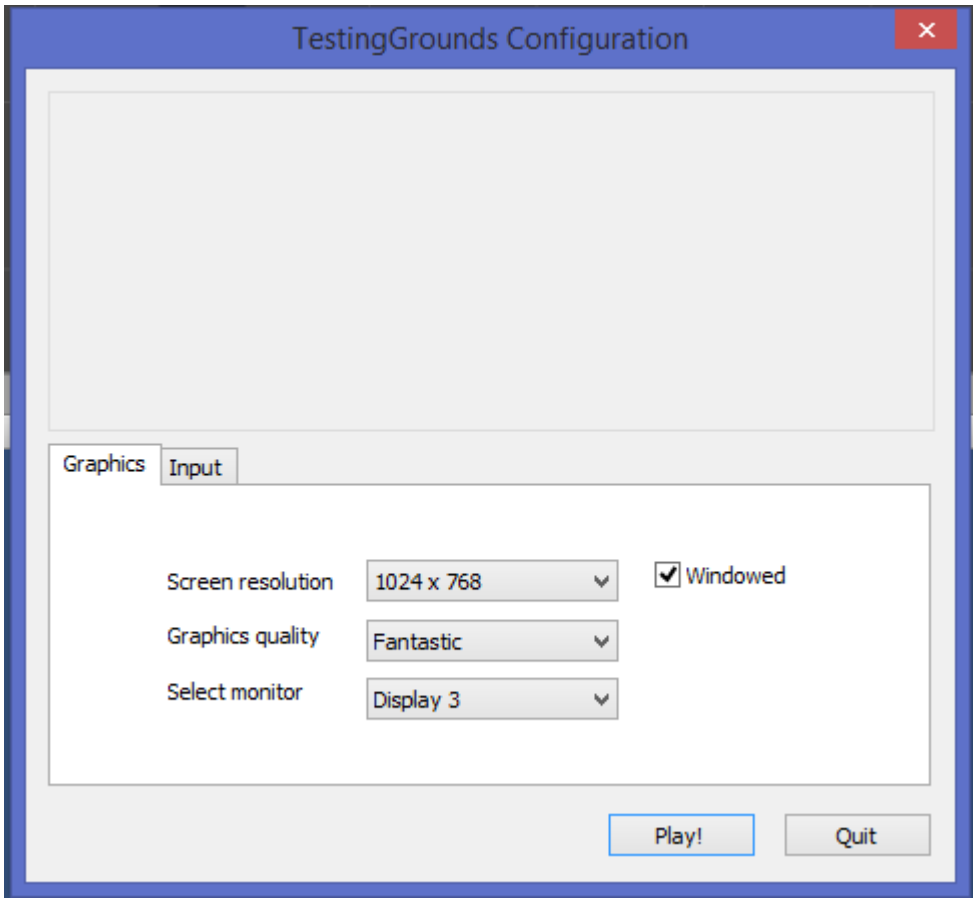

Figure 88. Unity resolution and quality window.

4.3. 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.

4.4. 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. Automatically, the editor window will be swapped with the game window.

Figure 89. Unity run controls.

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

4.5. 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!