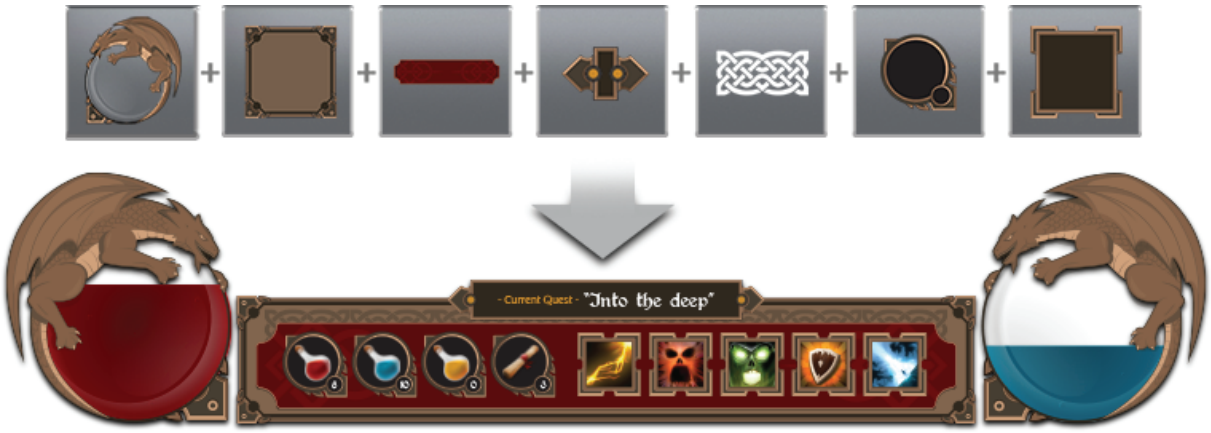
HUD & GUI MEDIEVAL ART BUNDLE

INTRODUCTION

Thanks for purchasing our pack **HUD & GUI Medieval Art Bundle**. This document is here **to help you understand how sprites are arranged along the pack and how to use them efficiently**. This pack uses separated sprites so you can combine them to create any type of interface you want. Below there is an example of a HUD made combining different sprites, but you can make any type of HUD you want, the limit is your imagination! This bundle also includes an example demo and example code and prefabs. You can use these prefabs and codes or create your own.



SECTIONS

Inside the pack you will find the following folders: **Buttons, Icons, Health, SFX and Windows**. The sprites are separated as single images and not spritesheets. This is done purposely so you can create your own spritesheets and have the best performance on your game. We recommend that you put in the same spritesheet only the images that are usually drawn at the same time and/or the same level or scene.

BUTTONS & ICONS

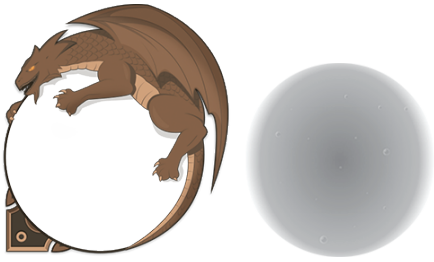
These folders contain many sprites to use as single images. Buttons are design to be used as clickable elements while icons are single images to use anywhere (such as gold rewards).

HEALTH

These images are made to represent numerical variables as bars. There are two types of healthbars: Rounded (as Diablo) and horizontal (as many other games like Street Fighter). The values that can be represented can be anything, like life points, magic points, experience points, or any mechanic implemented into your game.

ROUNDED HEALTHBARS

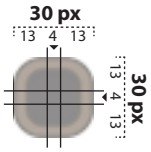
To use Rounded healthbars you must use a mask to hide the empty area. This mask will be moving while the value changes (at 100% the mask won't hide anything, and at 0% it will hide everything). A mask is an image that will make everything beneath it invisible.

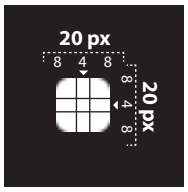


HUD & GUI MEDIEVAL ART BUNDLE

HORIZONTAL HEALTHBARS

To use Horizontal healthbars you must slice the bar sprite and the background sprite at 13 px from left and right. This will divide your image into three. You must scale the center of the sprite to represent the value, so at 100% the image will have the same size as the background, while at 0% the size of the bar will be 0.



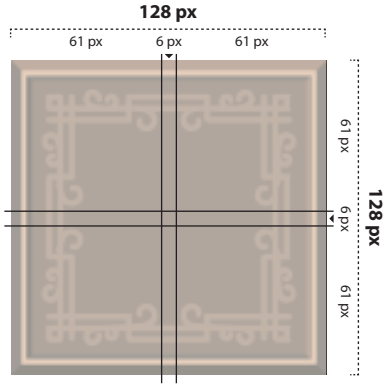


WINDOWS

Windows are images that are intended to be sliced, so they can be resized. They are intended to contain multiple elements. In this pack we have two different types of windows: background and ribbons.

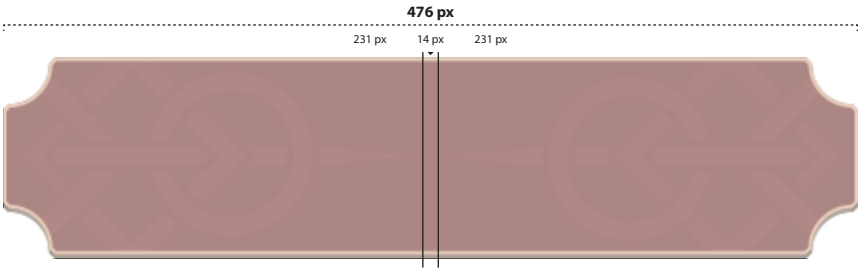
BACKGROUNDS

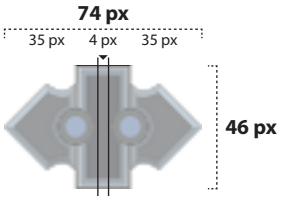
Backgrounds are the same size (128x128 px) and they are sliced at the same point (61 px from the edges). As a result, you will have 9 images: 4 corners, 4 sides and center. You can scale sides and center to adapt it to the size of the window, keeping the corners at the same size.



RIBBONS

Ribbons are sliced only from the sides, so you can't scale them vertically. These images are sliced from the left and right at 231 px. There is one exception thought, the image from the right is sliced at 35 px from left and right.





DEMO

There is a demo Scene under Demo/Scenes folder. This demo contains some examples of what you can achieve with this pack. The font used on the demo is the default Unity "Arial". We recommend to use these fonts for better results ***(click on the name to go to the link)***:

Augusta (http://www.dafont.com/es/augusta.font)
Centabel (http://www.dafont.com/es/centabel.font)
Germania (http://www.dafont.com/es/germania.font)
Sherwood (http://www.dafont.com/es/sherwood.font)
Vinque (http://www.dafont.com/es/vinque.font)

It is recommended that you do not include this demo on your builds to improve your build size. You can even delete this folder, but if you do, keep in mind that the example prefabs and example code are also on this folder. You choose if you want to remove or keep these prefabs and scripts to use in your project or not. Of course, you can use these prefabs and scripts on your game, but we recommend that you use your own for a better customization.

EXAMPLE PREFABS

- **ButtonText:** A button that switches its sprite to another one when the player clicks on it.
- **CastingRoundedButton:** This prefab is a button that when it is clicked it is disabled for a certain time and then enabled again. The button is animated while being disabled to tell the player how much time remains for it to be enabled again.
- **CastingSquaredButton:** Same prefab as the above but with a squared image instead of a rounded one.
- **Checkbox:** An example of a Unity UI "Toggle" with the graphics of the pack.
- **RoundCheckbox:** Another example of a Unity UI "Toggle" with the graphics of the pack, this time with the rounded graphics.
- **GathererLayout:** A prefab that draws an array of elements on the screens and enables a certain quantity of them (disabling the others). This is useful, for example, if the player has to collect 5 objects exactly. At the beginning there will be 5 disabled elements, but when the player collects one of the objects, the first element will be enabled. See the code section of "GathererLayout" for details on how to use this prefab properly.

- **HealthDragon:** A life bar with the Dragon sprites of the pack. Use the "Fill Amount" value of the image to change the amount of life remaining.
- **Slider:** An example of a Unity UI "Slider" with the graphics of the pack.
- **LifeBar:** Another example of a Unity UI "Slider", but representing a Life bar (so it has no handle and is not Interactable).
- **RoundedImageButton:** An example of a button with a rounded image of the pack. When the button is disabled, it changes the sprite to the "Locked" one.
- **SquaredImageButton:** Same as RoundedImageButton, but with a squared sprite.

EXAMPLE SCRIPTS

BUTTONDISABLER

This script changes the sprite of a button depending on if it is enabled or disabled. To use it, attach this script to your button and set the EnabledImage and DisabledImage, then, to change if the button is enabled or not, you must call "SetEnabled(true/false)" of this script for it to take effect. If you want to test this on the editor (even without the scene on play), you can click this icon on the component and select Enable Button or Disable Button.

BUTTONIMAGEONCLICK

This script changes the sprite of an Image when it is clicked. This script has 2 methods OnPointerDown and OnPointerUp to change the sprite. For these methods to be called, we need to add a EventTrigger component to the image, and add the PointerUp and PointerDown and subscribe these methods of the script. If you need more info on EventTrigger, see this page: http://docs.unity3d.com/ScriptReference/EventSystems.EventTrigger.html

CASTANIMATIONBUTTON

This class disables a button and enables it again after some time has passed. While it is disabled, it creates an animation modifying the fillAmount of an image to tell the user how much time remains for the button to be enabled again. The CastTime variable is the time for the button to be enabled again.

For this script to work properly, the button must be arranged as the prefab CastingRoundedButton, this is, the image of the button must be the locked one, and it must be set to filled. The button then must have a child with the enabled image. To start the animation, call StartAnimation() of this script.

FILLBYSLIDER

Example class used on the demo scene that sets a fill amount value depending on the slider value. To use this, add this component to the image. Then on the slider, add an "OnValueChanged" action, call to the "SetValue()" method of this class and attach the slider itself.

GATHERERLAYOUT

This class draws a specific number of sprites inside a layout and sets them enabled or disabled depending on another number representing the quantity enabled.

The **TotalElements** variable represents the total number of sprites that will be drawn, while **CurrentQuantity** is the number of sprites that are enabled.

To modify this values you must call the methods "SetTotalElements()" and "SetCurrentQuantity()" respectively.

SFX

Additionally, we included and extra 50 Sound Effects that you can use with your interface. These SFX are made by us if you can't find what your game needs take a look at our extensive high quality audio pack catalogue at **http://www.evil-mind.com/en/audio/**