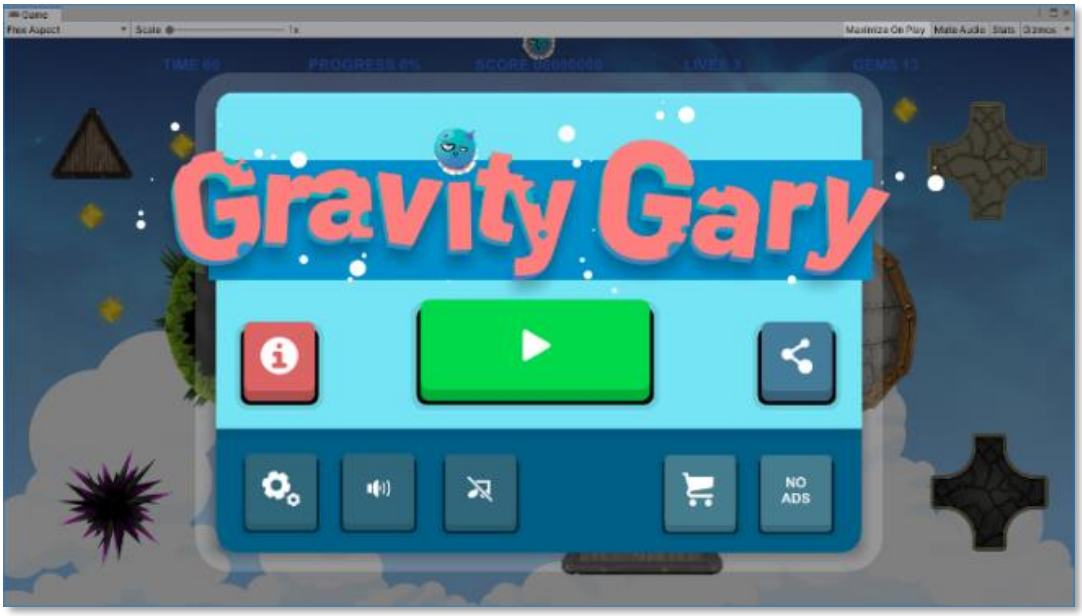
THE SIMPLE GAME UI

Documentation version 1.0

Questions about this product? Send an email to support@playniax.com



IMPORTANT NOTE! The demo uses a Test Game for loading. This probably won't work out of the box if you purchased the SimpelGameUI from the Unity Asset Store. You will need to add the Test Game (Test Level 1) to your Build Settings first (Scenes In Build).

INTRODUCTION

Probably every game out there has at least a loading, home, pause or game over screen.

To handle the different screens or game states, we need a proper manager who can provide a mechanism to know when to change from state "A" to state "B" during gameplay.

The SimpleGameUI is such a thing.

The SimpleGameUI provides a persistent class, meaning that it is created and it will persist throughout the game session, even if you switch scenes.

The SimpleGameUI will manage the different screens and can hold game data that has to be maintained throughout the game session like number of lives, current level, purchases, etc.

FEATURES

- Scene loading progress, pause and game over state etc.
- Buttons for turning music and sound on / off by the player.
- An info screen with scroll box for sharing instructions, info about your company or how to play the game etc.
- In-app purchasing button (removing ads).
- Unity Ads integrated.
- Play, exit or restart game buttons.
- 'Smart' music player.
- UI helper scripts for displaying player score, lives and current level
- Game timing helpers.
- Test keys for debugging.

FUTURE PLANS (SHORT TERM)

- A Level Selector screen
- Transitions

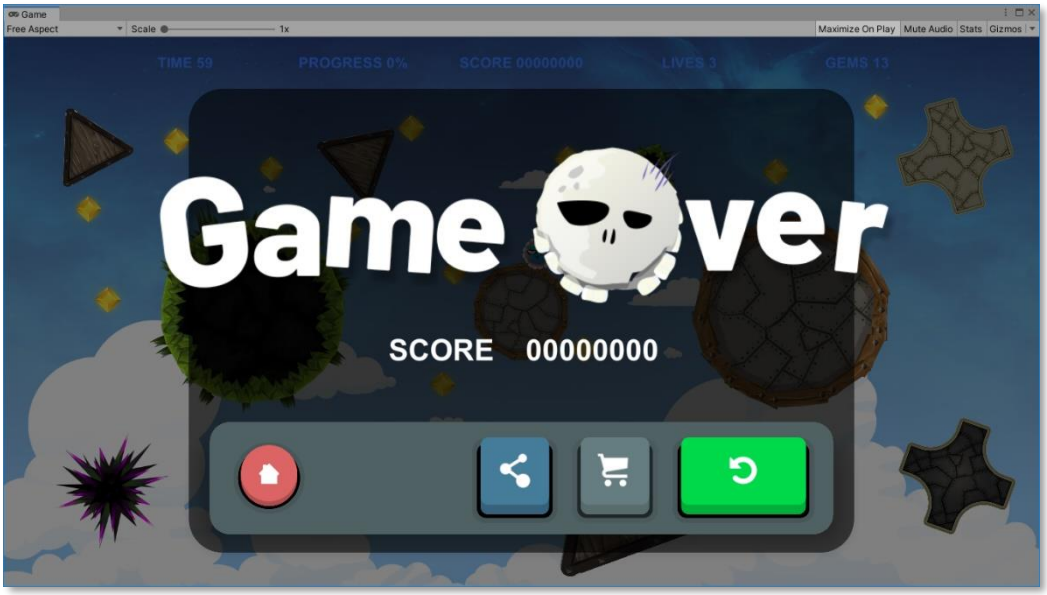
RE-SKINNING

The SimpleGameUI can be used straight out of the box. There are 2 generic templates for you to choose from. Just add your scenes to the Level Settings and the Level Manger will do most of the rest (don't forget to add the SimpleGameUI and the scenes you wish to load to the Unity Build Settings or it won't load).

The SimpleGameUI can be found in the 'Playniax / SimpleGameUI Themes' project folder. A test level can be found in 'Playniax / SimpleGameUI / _Test Level'. The test level contains an example script called 'Simulator.cs' and shows you how to get to the next level, game over screen and how to keep scores and lives count.

Always start off with a copy of a template for reskinning or change settings. This way you always have the original. The SimpleGameUI once loaded stays in memory or runs in the 'background' of your game (it uses DontDestroyOnLoad).

The SimpleGameUI should be the first scene to start in your build.


Game
Free Aspect
Scale
1x
Maximize On Play
Mute Audio
Stats
Gizmos
TIME 59
PROGRESS 0%
SCORE 00000000
LIVES 3
GEMS 13
Game Over
SCORE 00000000

SimpleGameUI (Script)

First Scene will be loaded when your app starts. Left blank the SimpleGameUI will load the first level or last level played depending on settings. Don't forget to add the scene to the Unity Build Settings or they won't load!

Start Paused - Your scene will start paused when Start Paused is enabled (TimeScale = 0).

New Game Every Time - Whether to reset the game or not on every play.

Advertisement Settings - Should be obvious. Basically an ad is shown between each level and new game.

Music Settings - Built-in Audio Source

Player Settings - Default number of Lives the player has.

Level Settings - The levels or scene to load as levels. Don't forget to add these scenes to the Unity Build Settings or they won't load!

Screen Settings - Contains some checkboxes to control loading behaviours and other UI settings.

There is also an option for choosing a Game Over and a Next Level key for simulating these states in the game. This is for debugging and the keys will not work in the final build.

IMPORTANT VARIABLES AND FUNTIONS

SimpleGameUI.instance - The instance of the SimpleGameUI.

SimpleGameUI.instance.playerSettings.lives - The number of lives the player starts with

SimpleGameUI.instance.Levelup() - Call this from your scene when the player finishes a level. The SimpleGameUI will load the next level.

SimpleGameUI.instance.GameOver() - Call this from your scene when the game is over. The SimpleGameUI will show the game over screen.

SimpleGameUI.instance.GetCurrentLevel() - Returns the current level active.

SimpleGameUI.instance.isLastLevel - Returns True is the last level is currently active.

PlayerData.Get(index).lives - Can be used to keeps track of how many lives the player has left. The LivesUI.cs script uses it to display the number of lives. Index represents the player e.g. 0 = player 1, 1 = player 1 etc.

PlayerData.Get(index).scoreboard - Can be used to keep track of how many points the player has. The ScoreboardUI.cs script uses it to display the score. Index represents the player e.g. 0 = player 1, 1 = player 1 etc.

Timing.Paused() - Returns True if game is paused (TimeScale = 0)

LICENSE (this basically sums it up)

- The template can be used to create non-commercial or commercial titles
- Credit / attribution is not mandatory, but appreciated
- The template and the assets itself cannot be shared freely or (re)sold!

CREDITS

- Programming by **T**ony **S**mits
- Graphics by **P**eter van **D**riel
- Music by Bensound

SERVICES

We provide additional payed services such as

- Designing a custom game or prototype for you
- Designing custom art
- Reskinning
- Personal support or Unity lessons to get you started

Contact us at support@playniax.com