Astro Bouncer Game Template

Game documentation and HowTo guide.

This document contains:

Package Description and features

Astro Bouncer Game is a full Unity template ready for release. It is compatible with mobile as well as standalone and webplayer.

How to Play?

Click or press Space to jump. Collect stars and avoid spikes!

Watch the gameplay video

Current version 1.4

Update history

1.4 (27.04.2018)
- Cleaned up project assets so they can be easily mixed into other projects.
- Removed tags that are no longer used for Enemy, Item, Floor, etc.

1.36 (10.12.2017) - Removed all JavaScript assets to prevent future compatibility issues.

1.34 (25.10.2017) - Support for Unity 5.5, 5.6, and 2017 1.32 (13.05.2016)

1.32 (13.05.2016)
- Support for versions of Unity higher than 5.3.
- Better UnityAds support for Unity 5.2 and above.

1.31 (22.12.2015)
- Added support for UnityAds along with a quick integration guide.
- Added support for Unity 5.3 SceneManager.
- Updated versions for Unity 4.6.9, 5.1, 5.2, and 5.3

1.28 (30.08.2015)
- Entire project ported to C#. The package now contains both versions for JS and C#.

1.18 (03.08.2015)
- Update buttons to use a unified function. This prevents errors when upgrading to higher versions.
- Made sure all sounds are CC0 compliant.
- Updated project to latest versions of Unity 4 and Unity 5.

1.17 (17.05.2015)
- Updated project to Unity 4.6.5 and Unity 5.

1.16 (15.03.2015)
- Changed in-game button input to MouseDown to improve gameplay responsiveness.

1.15 (10.03.2015)
- Added 3 new characters you can choose from in the main menu.
- Added a new floater enemy, which chases the player and can be killed by bouncing on it.
- Reorganized controls; Player Controls and Move Area are now placed in the GameController script to allow for easier multiple characters.
- Reorganized the game area in the scenes to better support multiple device sizes.
- Minor changes.

1.1 (27.01.2015)
- Added a new game mode where the player can move left/right, and go all the way from one side of the screen to the other.
- Added bouncing spikes that come up from the bottom.
- Added a TextByPlatform script which allows the user to customize a text string based on the platform type.

Credits

The sounds are courtesy of the free sound project.

Intro music is Electronic U Popper by Frank Nora

Credits go to these authors for their great sound samples: **Tristan, captainpugwash, fins, cylon, eaglestealthteam**

Please rate my file, I'd appreciate it 🙂

Overview of the game's library contents

Let's take a look inside the game files. Open the main AstroBouncerAssets folder using Unity3D 4.6 or newer. Take a look at the project library, usually placed on the right or bottom side of the screen. Here are the various folders inside:

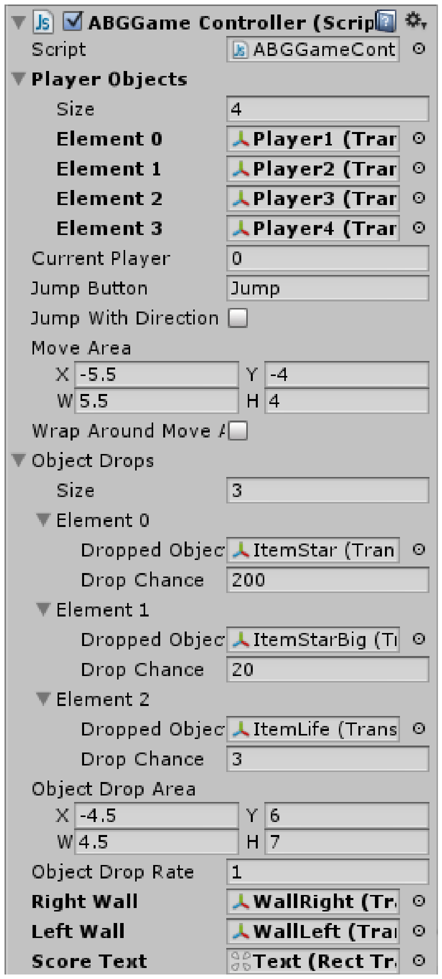
- **Animations:** Holds the animation clips made with Unity’s built-in animation system.
- **FLA:** Holds the object graphics made with Flash CS3. These are vector graphics than can be easily scaled without loss of quality and then exported as PNG to be used in Unity.
- **Fonts:** Holds the font used in the game, AGENCYB.
- **Prefabs:** Holds all the prefabs used in the game. These are distributed to various folders for easier access, Buttons, Enemies, Objects, etc
- **Scenes:** The first scene that runs in the game is MainMenu. From this scene you can get to the Game scene.
- **Scripts:** Holds all the scripts used in the game. Each prefab contains one or more of these scripts.
- **Sounds:** Holds all the sounds used in the game. Jump,Item, etc
- **Textures:** Holds all the textures used in the game which are used as sprites in Unity.

Getting started

Astro Bouncer Game (ABG) is considered a complete project, and as such is supposed to work as the starting point of your planned game, rather than an addition to an existing project. That said, you may of course pick and choose some of the scripts/models to import into your existing project, but ABG works best as a starter kit which you can customize any part of to your liking.

The Game Controller

The Game Controller is the main prefab that controls all the progress of the game from start to finish. It controls the UI of the game, spawns items and enemies, and also shows/hides the spikes on the walls. The Game Controller is also used to increases the difficulty of the game after collecting a set amount of points.



```
ABGGame Controller (Scrip
Script                  ABGGameCont
Player Objects
  Size                  4
  Element 0             Player1 (Trar
  Element 1             Player2 (Trar
  Element 2             Player3 (Trar
  Element 3             Player4 (Trar
Current Player          0
Jump Button             Jump
Jump With Direction     ☐
Move Area
  X -5.5                Y -4
  W 5.5                 H 4
Wrap Around Move A      ☐
Object Drops
  Size                  3
  Element 0
    Dropped Objec       ItemStar (Tran
    Drop Chance         200
  Element 1
    Dropped Objec       ItemStarBig (Ti
    Drop Chance         20
  Element 2
    Dropped Objec       ItemLife (Trans
    Drop Chance         3
Object Drop Area
  X -4.5                Y 6
  W 4.5                 H 7
Object Drop Rate        1
Right Wall              WallRight (Tr.
Left Wall               WallLeft (Trai
Score Text              Text (Rect Tr.
```

Player Objects – Holds the list of players that can be in the game. At the start of the game the number of the current player is loaded from PlayerPrefs, and all other player objects are deactivated.

Current Player – Is the number of the current player.

Jump Button – Is the button that makes the player jump. This is defined in the Input Manager.

Jump With Direction – Should the player be able to jump left/right? To make this work you must assign an axis in the Jump Button field, such as "Horizontal".

Move Area – The area in which the player can jump.

Wrap Around Move Area – Should the player bounce off the sides of the move area or should it wrap around them (teleport from one side to the other when going off screen).

Object Drops – Holds the list of items and enemies that can be dropped during the game. Each object has a drop chance. The drop chances are calculated relevance to each other. For example, if a list contains object A with a drop chance of 1 and object B with a drop chance of 5, it means that the object B will drop 5 times more often than object A.

Object Drop Area - This is the Rectangle within which objects are dropped.

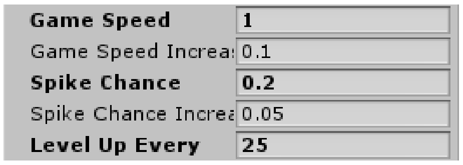
Object Drop Rate – This is the number of seconds to wait before dropping another object.

Right/Left Wall – These hold the wall objects on the sides of the screen, which have spike enemies inside them. The Game Controller shows/hides the spikes randomly when the player bounces off a wall.

Score/Score Text – Holds the current score and the text in which it is displayed.

Changing Game Difficulty

The GameController also allows you to change the difficulty of the game through several variables, making the game get gradually faster with more obstacles as you rise in the levels.



Game Speed	1
Game Speed Increa:	0.1
Spike Chance	0.2
Spike Chance Increa	0.05
Level Up Every	25

Game Speed – Is the current speed of the game.

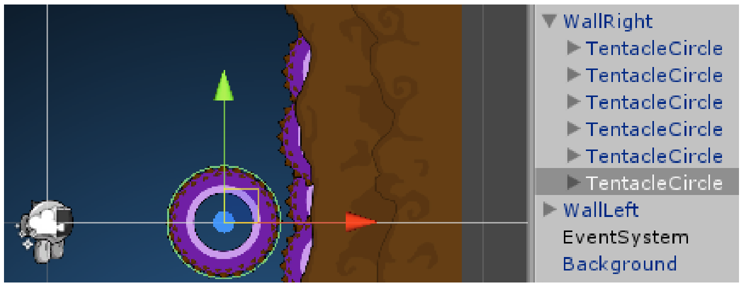
Game Speed Increase – Is the change in speed with each new level. Example: If we start at a **Game Speed** of 1, and increase the speed by 0.1 with each level, we would be at **2** speed after 10 levels.

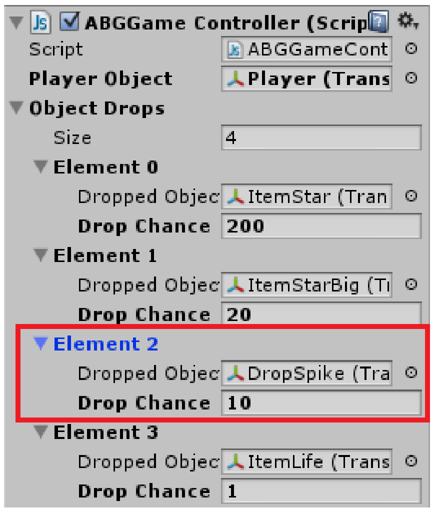
Spike Chance – The chance for a spike to appear in the wall. Even if the chance is 1 (meaning that the spike always appears), there will always be one spike hidden in the wall to allow the player a chance to avoid being hit.

Spike Chance Increase – The increase in the chance of a spike to appear in the wall, similar to how **Game Speed Increase** works.

Level Up Every – How many points the player must collect in order to rise a level. When the player rises a level, the **Game Speed** and **Spike Chance** increase.

You can also increase the number of spikes inside the walls. Simply clone any of the spikes inside and place them where you want.





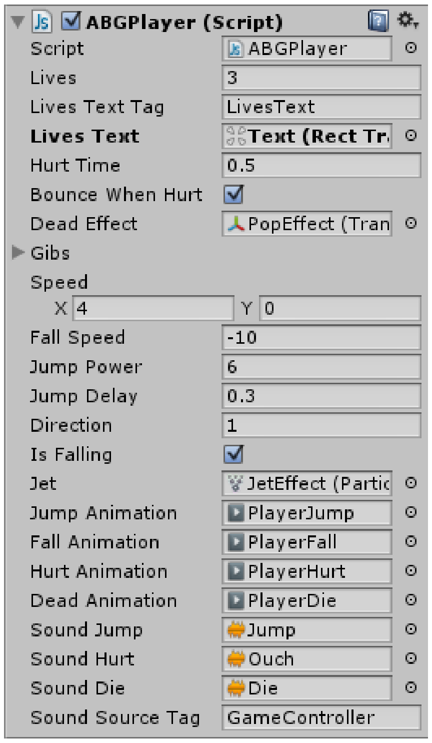
ABGGame Controller (Scrip

Script	ABGGameCont
Player Object	Player (Trans
Object Drops	
Size	4
Element 0	
Dropped Objec	ItemStar (Tran
Drop Chance	200
Element 1	
Dropped Objec	ItemStarBig (Ti
Drop Chance	20
Element 2	
Dropped Objec	DropSpike (Tra
Drop Chance	10
Element 3	
Dropped Objec	ItemLife (Trans
Drop Chance	1

Another way to increase overall difficulty in the game is to change the drop chance of the Spike ball that is dropped from the top of the screen.

Editing the Player

The player object has several variables that can change how it behaves, mainly the jump power and fall speed, and of course the health of the player.



ABGPlayer (Script)

Script	ABGPlayer
Lives	3
Lives Text Tag	LivesText
Lives Text	Text (Rect Tr
Hurt Time	0.5
Bounce When Hurt	✓
Dead Effect	PopEffect (Tran
Gibs	
Speed	X 4 Y 0
Fall Speed	-10
Jump Power	6
Jump Delay	0.3
Direction	1
Is Falling	✓
Jet	JetEffect (Partic
Jump Animation	PlayerJump
Fall Animation	PlayerFall
Hurt Animation	PlayerHurt
Dead Animation	PlayerDie
Sound Jump	Jump
Sound Hurt	Ouch
Sound Die	Die
Sound Source Tag	GameController

Lives/Text – The number of lives the player has, and the text in which they are displayed.

Hurt Time – The time when the player is hurt, and becomes invulnerable and uncontrollable.

Dead Effect – The effect object that is created after the player object is removed. In our case, it's the Pop effect after the player explodes.

Gibs – This is a list of gibs that will fall off the player when it is destroyed. We assign these from the player itself and they simply detach from it when it dies.

Speed – The current horizontal and vertical speed of the player.

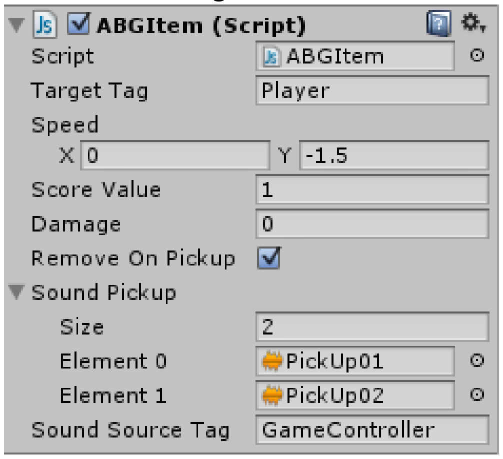
Fall Speed – The gravity of the player. We didn't use Unity's default gravity because we want to have better control over our game.

Jump Power – How high the player can jump.

Direction – The current horizontal movement direction of the player.

Editing Items

Items are spawned by the game controller, and can be picked up by a target object (usually the Player). An item can change the health of the target and the score in the game.



ABGItem (Script)	
Script	ABGItem
Target Tag	Player
Speed	
X	0
Y	-1.5
Score Value	1
Damage	0
Remove On Pickup	✓
Sound Pickup	
Size	2
Element 0	PickUp01
Element 1	PickUp02
Sound Source Tag	GameController

Target Tag – Is the name of the tag of the object that can pick up this item.

Speed – The vertical and horizontal speed of the item.

Score – How many points we get when this item is picked up.

Damage – The change in health of the target that picks up this item. You can make an item that increases lives by setting the number to negative.

Remove On Pickup – If set to true, this object will be destroyed when picked up.

Sound Pickup – A list of sounds that will be played randomly when this object is picked up.

Sound Source Tag – The tag of the source object from which the sounds play.

UnityAds Integration (Unity 5.2 +)

Since Unity 5.2 UnityAds integration has been simplified, here's how you can have full screen video ads in your game.

This video shows a quick process of integrating UnityAds into your project. In the example we used one of my templates, but it works on all my other templates too.

https://www.youtube.com/watch?v=EQNTgfV35DU

Here is what we did in the process:

1. Sign in to your Unity account in order to allow Unity Services such as UnityAds to be activated.
2. Open Build Settings and switch the platform to one of the supported ones (iOS, Android).
3. Download Puppeteer's UnityAds package from: https://drive.google.com/file/d/16NBhz2hVlThHXjkZ0k1z6BXPAhHfItZM/view?usp=sharing
4. Drag the downloaded package into your Unity project, and import it. This UnityAds prefab can be used to display ads every several minutes.
5. Drag the prefab into any scene where you want ads to be shown. Make sure to save changes.
6. The time check is shared between all prefabs in all scenes, so you will never show too many ads.
7. The final step is to activate UnityAds services and get your unique project ID.
8. Open the services window and choose your organization, then click create.
9. Choose UnityAds from the list and turn it On.
10. Choose age group for your project (Will affect the nature of ads shown), and save changes.

11. While working on your project keep Test Mode activated. But when you are ready to release the final project, switch Test Mode off.

12. That's it! Now when you start the game, an ad will be shown after 3 minutes. The ad will never appear during gameplay or post-game screen. Instead, it will wait until the next level load (restart, main menu, etc) and then show the ad.

Before releasing a game, make sure you uncheck **Enable Test Mode.**

For more info about integrating UnityAds read this:

http://unityads.unity3d.com/help/monetization/integration-guide-unity