KrazyBall Level Creation Guide

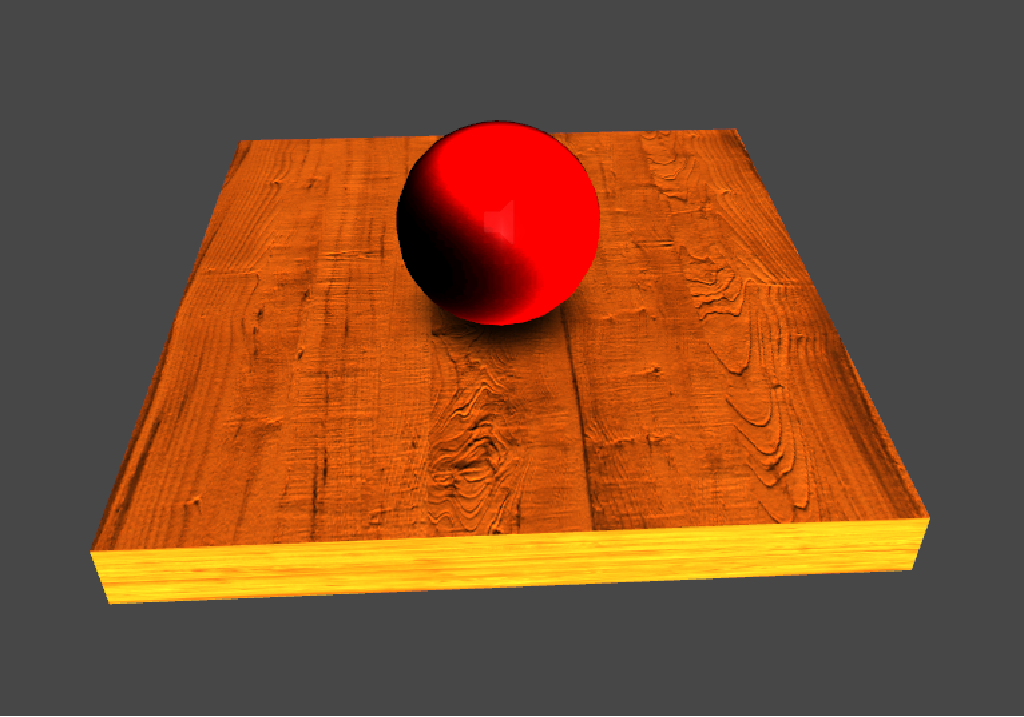


Table of Contents

[Foreword 3](#_Toc370579171)

[Prefabs 3](#_Toc370579172)

[Level Essentials 3](#_Toc370579173)

[Ball 3](#_Toc370579174)

[Exit 4](#_Toc370579175)

[MainCamera 4](#_Toc370579176)

[MainLight 4](#_Toc370579177)

[Menu 4](#_Toc370579178)

[Win 4](#_Toc370579179)

[Game Elements 4](#_Toc370579180)

[FLOORtag, noFLOORtag 4](#_Toc370579181)

[boost, booststrong, boostweak 5](#_Toc370579182)

[large, medium, small, largeonce, mediumonce, smallonce 5](#_Toc370579183)

[obstacle 5](#_Toc370579184)

[switch, switchOnce 5](#_Toc370579185)

[teleport 6](#_Toc370579186)

[spikes 6](#_Toc370579187)

[enemy 6](#_Toc370579188)

[spring 6](#_Toc370579189)

[smash 6](#_Toc370579190)

[Memory Game 7](#_Toc370579191)

[ARROWMEM 7](#_Toc370579192)

[MEMBLOCK 7](#_Toc370579193)

[MEMSTOP 7](#_Toc370579194)

[Geometry Models 7](#_Toc370579195)

[Pipe 7](#_Toc370579196)

[Pipe\_spike 8](#_Toc370579197)

[Pipe\_wall 8](#_Toc370579198)

[Ramp 8](#_Toc370579199)

[Other 8](#_Toc370579200)

[Loadlevel 8](#_Toc370579201)

[NewGame 8](#_Toc370579202)

[ProgressUnlock 8](#_Toc370579203)

[Quit 8](#_Toc370579204)

[Reset 8](#_Toc370579205)

[Tips 9](#_Toc370579206)

[ArrowBlack 9](#_Toc370579207)

[Animation 9](#_Toc370579208)

[Shadows 10](#_Toc370579209)

[Conclusion 10](#_Toc370579210)

# Foreword

This purpose of this brief guide is to explain and hopefully teach potential level creators how to use the KrazyBall Unity3d project to make their own levels and even share them with others! When I developed KrazyBall, I specifically made sure to create the tools and functionality that would enable me to make levels without writing a single line of code. This made the level making process much more enjoyable for me personally because I’m not really that crazy about coding all the time, and as a side bonus anyone else with no coding experience whatsoever can now make levels. You just have to know how to use the Unity3D visual editor and understand how the building blocks of KrazyBall work, which this guide elaborates on.

# Prefabs

## Level Essentials

The level essentials folder contains the prefabs that EVERY level MUST contain. There are six of them and they are:

* Ball
* Exit
* MainCamera
* MainLight
* Menu
* Win

### Ball

The ball prefab is the player character. It’s obviously necessary in every level or else the user would have nothing to control. Wherever you place the ball in the level is where it will be when the level starts. The **Deathheight** variable located in the Balllogic component controls the Y-value at which the level will restart if the ball reaches it. You can change it if it’s too high and your level goes deeper beyond it, or of course you could just make your level higher. I like to always place the ball object at position 0,0,0 when making levels, so for example in level 10, I set **Deathheight** to -500 instead of the default -100. Only one ball object is allowed.

### Exit

The exit prefab is the level exit that the ball must make contact with in order to win that particular level. In the Win Script component, the **Win** variable must contain a reference to the Win prefab that must also be present in every scene and is discussed last in the level essentials section. The **Next Level** variable should contain the value of the level after the current one. If the level currently being played is level 1, then the **Next Level** value is 2. This is to ensure that the next level is unlocked in the hub level. You can have multiple exits in a level.

### MainCamera

This is the custom main camera for the game. When you create a new scene, it always contains a default MainCamera. Delete that one and replace it with the MainCamera prefab. Afterwards look at the **Target** variable in Mouse Orbit component and the **Target** variable in the Camera Object Fade component. Give both of them a reference to the ball prefab in the scene. That’s it. Now even if you move the camera around while making the level, when the level begins it will snap to the ball correctly wherever it is. Only one MainCamera is allowed.

### MainLight

You can just this drop into the scene and forget it. It creates the default lighting like in all the demo levels. Only one MainLight is allowed.

### Menu

On contact with the ball prefab object, the menu will load. You can have multiple menu objects in a level.

### Win

Just drop into the scene and forget it, but make sure the x and y position values are both 0.5, and make sure the **Win** variable in the Exit object references this Win object.

## Game Elements

Gameplay elements are the meat of KrazyBall. They are the main tools a level creator uses to make unique and compelling experiences for the player. All the objects in the GameElements folder will be discussed here, except for the Animation subfolder which is discussed later.

### FLOORtag, noFLOORtag

FLOORtag means floor, and a noFLOORtag means wall. The ball reacts to them differently. The ball only ‘rolls’ on floors and not walls so there are a few basic rules you must always follow when using them. A wall must ALWAYS be vertical. If a ball can somehow land on top of it in a level, then you must place a floor on top of it so the ball can properly roll. All ten demo levels follow these rules. If you look at level 9, the moving vertical walls have thin floors on top of them in case the balls fall from above. It’s good to be consistent with the thickness of the connected floors and in level 9 they all have a thickness of 0.02(as child objects).

### boost, booststrong, boostweak

As their naming suggests, these are three boost platforms with varying boost strength. Level 5 starts with a booststrong and the the one in front of the ramp is a regular boost, while level 9 uses a bunch of boostweak platforms. The ball is boosted in the direction of the arrow. Works on enemies too!

### large, medium, small, largeonce, mediumonce, smallonce

These are size changing objects. Both the player and enemies are affected by them. Large, medium, and small are reusable platforms, while largeonce, mediumonce, and smallonce are one time pickup objects. Fun fact: enemies that are the same size or large then the player are dangerous, but smaller ones are destroyed on contact.

### obstacle

This is just a floor with a different texture. I use it to give the player a visual clue that it’s meant to be crossed over by a ‘large’ ball at some point, such as in level 1,2,6, and 7.

### switch, switchOnce

Switch is toggle object while switchOnce is a onetime switch. A switch is customized using the Switch or Switch Once component. The component has four sections:

* To Disable

Objects referenced here are disabled when switch is touched.

* To Enable

Objects referenced here are disabled when switch is touched.

* Animation Disable

Objects referenced here will have their animations paused when switch is touched.

* Animation Enable

Objects referenced here will have their animations unpaused when switch is touched.

Keep in mind that regarding toggle switch contact, objects that have been disabled would be once again enabled and vice versa. The same principle applies to animations. Both players and enemies can interact with switches.

Also, in level 2 there is a blue and a red switch as well as a red and blue wall. This was done by dragging the SwitchRed and SwitchBlue materials from the Materials folder onto the switches. The same thing was done for the walls using the WallBlue and WallRed materials. The reason I did not make these into separate prefabs is because there could be infinite different colors that can be used for switches, walls, etc, and all you need to do is make a new material for each color, which is super easy.

### teleport

The player or enemy will be teleported on contact to the 3D location specified in the **Spawnpos** variable in the Teleport component. Velocity and spin are conserved.

### spikes

Player contact with this object will restart the level. Enemy contact will destroy the enemy.

### enemy

The enemy can interact with any object the player can. It has a detection sphere around itself and will lunge at the player if the player enters the sphere and the enemy can see the player. If the enemy collides with any object too fast then it gets stunned for a brief time. During that time it cannot harm the player in any way and will not attack but a larger player cannot destroy the enemy either for the duration of the stun.

### spring

Springs apply a force on a player or enemy in the direction of the spring’s face. The force depends on the vertical velocity with which the player or enemy collides with the spring. This means if the ball is going super fast horizontally and grazes the spring face then it won’t bounce off super fast. Keep in mind that a ‘large’ ball will slow down more and more while a ‘small’ ball will keep speeding up. If for the ‘small’ ball the bounce isn’t stopped eventually, the ball will go so fast that it’ll go through objects. In level 10, I solve this problem by placing spikes on the ceiling above the spring, so if the ‘small’ ball goes too high and collides with it then the level will restart before things get out of hand.

### smash

Smash objects work in pairs. They are intended to be placed in spots where the ball can get crushed such as between a moving platform and a fixed platform in level 6. Basically if the player or enemy is in contact with 2 or more smash objects at the same time, the level restarts or the enemy is destroyed respectively.

In level 10, the smash object on the floor under the square block wall is much wider than the square block itself and the smash object attached to it. This is because if the player is ‘large’ then a piece of the ball will be between the block and the floor before it comes into contact with both smash objects and not get destroyed when the block starts crushing it. By making the smash object on the floor wider we can ensure that the large ball will definitely be in contact with it when the block starts crushing the ball.

## Memory Game

This part pertains to the “remember the path” gameplay as seen in level 3 and level 10. The 3 main prefabs relating to this are:

* ARROWMEM
* MEMBLOCK
* MEMSTOP

### ARROWMEM

These arrow objects do not interact with anything. Their purpose is to visually show the path that needs to be remembered.

### MEMBLOCK

This is one of the main components of the memory game. These objects are invisible and deactivated when a level begins. This prefab is meant to be placed everywhere the player is NOT meant to go once the memory game begins. A player begins a memory game by activating a onetime switch that contains references to the relevant MEMBLOCK in the **To Enable** array of the Switch Once component. Once the objects are enabled, they are still invisible but player contact with them will cause the level to restart. Use these in combination with ARROWMEM objects to create a functional memory game.

### MEMSTOP

This object can be used to stop a memory game. It should be enabled with the MEMBLOCKs once the switch is hit. It acts as a onetime switch on player collision, so any enabled MEMBLOCKS referenced in the **To Disable** array in the Switch Once component will be disabled. You can also place something in the **To Enable** array such as a missing bridge that appears once you have beaten the memory game.

## Geometry Models

All the prefabs in the Geometry Models folder will be discussed here except for the pipeplank prefab which will be discussed later in the shadows section. These prefabs are:

* Pipe
* Pipe\_spike
* Pipe\_wall
* Ramp

### Pipe

The Pipe prefab is basically a FLOORtag object in the form of a pipe. It’s actually 2 pipes in one but I will explain why that is in the shadows section. It can be placed in any way and it’s used in level 1,2, and 6.

### Pipe\_spike

This is just the spike prefab in the form of a pipe to be wrapped around pipe objects.

### Pipe\_wall

The Pipe\_wall prefab is basically a noFLOORtag object and should only be placed vertically like all noFLOORtag prefabs, such as in level 10. The floor part of the object should be faced upwards in case the player lands on top of it.

### Ramp

This is a FLOORtag object with a slippery surface. This allows for the ball to easily accelerate up and down the ramp, especially when boosted onto it such as in level 5 and 8.

## Other

The following objects are mainly used only in the hub level:

* Loadlevel
* NewGame
* ProgressUnlock
* Quit
* Reset

### Loadlevel

Loads the level specified by the value in variable **Load Level** in the Load Level component. Level values are determined in the build settings of the project. Keep in mind that the level will only load if it’s unlocked by the exit object of another level. This is how progression works in KrazyBall.

### NewGame

Upon contact, the save game value in the registry is either reset or is newly created, effectively starting a new game. The level restarts as well.

### ProgressUnlock

If the **Progress Level** variable in the Progress Unlock script is equal to or less than the global level progress variable, then the objects in **To Enable** and **To Disable** are enabled and disabled respectively when the level loads. This is used in the hub level to load the YOU WON THE DEMO!!! text with secret jedi mind trick donation message.

### Quit

Game exits upon player contact.

### Reset

Similar to the NewGame prefab, except instead of resetting the level progress variable, it gets removed from registry. I know the name is counterintuitive lol.

## Tips

### ArrowBlack

Use to help with direction. No interaction at all.

# Animation

I’ll try to keep this section short and sweet. There are two prefabs In the Animation subfolder located In the GameElements folder. They are:

* movingFloor
* movingWall

INSIDE the 2 prefabs, there is an identically named child object. The purpose of this is to drastically reduce the number of animation clips that need to be made for any particular level. Basically, the inner child object contains all the functionality while the parent object is just a transform component as can be seen by clicking on it. How this works is best illustrated by example using level 9.

Look at the first three sliding walls that the player must get by, which are called slidingWall1, slidingWall2, and slidingWall3. slidingWall1 is the ORIGINAL object and it uses animation clip **level9updown**. We know it’s ORIGINAL because the parent object has position (0,0,0) and rotation (0,0,0). Now look at slidingWall2 and slidingWall3. The inside child object **movingWall** is identical for all 3 slidingWall objects, BUT the parent objects have different values for position and rotation! slidingWall2 has position x = -42.07753 and rotation y = 180, while slidingWall3 has position x = -30. By adjusting the parent position and rotation it is possible to just make copies of the original animated object and move it around without creating new animation clips for every situation. Instead of three different animations for the three sliding walls, I only needed one. This is the approach I used for all the “similar looking” animated objects in level 9, such as the smashing cubes, and rotating floors.

There is one more thing to know about animations. While there is only a movingFloor and movingWall prefab, any other object can be attached to them to make them animate as well. Looking once again at level 9, the first 3 sliding wall objects have thin FLOORtag objects attached to them. More specifically, the floors are attached to the CHILD movingWall inside the parent movingWall which is only a transform as mentioned earlier. After the 3 sliding walls there are 2 crushing cubes which have a floorTAG AND a smash prefab attached to the CHILD object. Therefore by attaching objects in that fashion, anything can technically be animated. You can even make the size of the movingFloor or movingWall (0,0,0) to only make it look like the attached object is there and not the floor or wall. I also try to be consistent with the attached floor sizes, so you’ll notice that in level 9, all the floors on top of the animated movingWalls have a thickness of 0.02. Consistency is really important when making your levels, so try to be as consistent as possible.

# Shadows

I have created a magical layer based shadow system that uses the shadow projector and prevents it from rendering shadows through surfaces. The following layers are defined in the layer section of the editor: level0 – level10, T01 – T9,10.

For example, if the player is directly above a FLOORtag that has layer level0, this means that the shadow projector will render shadows on ALL surfaces that are in layers level0 AND T01. If the FLOORtag has layer level1, then shadows would be rendered on surfaces that are in layer level1, T01, and T12. The T layers are there to provide smooth transitions between surfaces with different layers and are used for the tube in level 2 and some FLOORtags in level 8.

When placing tubes, I also place the floorpad prefab at the end of each side of the tube. I mostly do it for consistency but in some situations it's necessary to prevent shadow clipping between layers. Also, tubes are actually made up of 2 tubes, an inner and outer one. See how the layers were set for the tube and floorpad in level 2 to prevent the shadow from rendering through the tube and to facilitate smooth shadow transition between floor, floorpad, and tube.

The only weakness of this implementation is that if there's no way to smoothly transition between 2 surfaces, then the shadow will clip to the next surface once the center of the ball reaches it while no longer rendering on the surface the ball was rolling from. Other than that, the system works very well and is implemented in every level of the demo.

# Conclusion

This guide covered all the prefabs available for level creation as well as animations and shadows. There is absolutely no coding required and any levels created can be easily shared by either continuing after level 10 or starting over from level 1 again. Of course all levels can be made available by setting levelprogress to whatever value you need when exiting any particular level. If anyone has any questions whatsoever, feel free to message me at my indiedb page: http://www.indiedb.com/games/crazy-ball.

I've worked on this project for over a year and learned a lot, so I'd be happy to teach or make a video series or something if anyone's interested.

## 

### 

### 