lòÓneyBITS

SuperGoalKeeper

05/28/2014
Update 05/31/2014 Variables desc.
Update 06/12/2014 Social Buttons.
Update 07/25/2014 Gloves Menu, Random field grass & 1 dynamic scene.
Update 10/28/2014 Added new scenes, game_shop & game_base.
Update 06/06/2015 Look&Feel updated and goal post sound effect added.
Update 11/11/2015 Look&Feel updated, new gloves and scenes added.

Is a production of Looneybits
http://www.looneybits.com

Table of Contents

1. Super Goalkeeper

1.1.Summary

This is a fun and useful "catch game" style game. This complete project will help you develop other games faster. Code is fully commented read to customize the game.

1.2.Document tree

The project has the following folders:

- Physics Materials: This folder contains the material of balls/coins.
- Music: Here is the sound effects and music.
- Prefabs: This folder contains the objectsGame(player, ball, goal...) of game.
- Scenes: This folder contains the scenes of the game.
- Scripts: This folder contains the C# scripts of the game.
- Sprites: This folder contains the sprites .
- Src: This folder contains SVG files.
- game_base: This folder contains libs of game.
- game_shop: This folder contains shop of game.

1.3.Variables of interest

Below there is a list with variables to customize how the game works.

Script/Class	Type	Name	Desc
MissionOne.cs	Camera	cam	Main camera.
	GameObject[]	**gloves**	**Gloves prefabs array. If you create a new glove prefab you should add here. Load the glove selected by user.**
	GameObject[]	**grassList**	**Field grass prefabs array. The machine choose randomly the grass field.**

	GameObject[]	**objectGame**	**Ball array or enemy array. Array that contains the objects that will fall.**
	GameObject	**grass**	**Terrain GameObject.**
	GameObject	**coin**	**Coin GameObject.**
	PlayerModel	**player**	**Is the player object.**
	GoalModel	**goal**	**Is the goal object.**
	bool	gameOver	Flag indicates the game is finished.
	float	**gameDuration**	**Game duration.**
	float	maxWidth	Indicates max width of game scene.
	float	**seconds**	**Increased seconds when player caught an object.**
	float	**penaltyTime**	**Penalty time.(seconds)**
	int	**balls4Cash**	**Number of balls to spawn coins.**
	int	**spawnNCoins**	**Number of coins to spawn.**
	Vector2	**timeRange**	**Balls spawner time range.**
	int	oldCollectedObjects	Old value of caught objects.
	int	oldScoredGoals	Old value of scored goals.
	bool	missionEnd	Flag indicates that mission is finished.
	int	oldCoins	Old value of caught coins.
	IndicatorView	iLevel	Level indicator.
	IndicatorView	iCrono	Time indicator.
	IndicatorView	iHealth	Health indicator.(NOT IN USE)
	IndicatorView	iCoins	Coins indicator.
	IndicatorView	iLevelUp	LevelUP inidcator.
	MenuView	gameOverMenu	Game Over Menu.
	MenuView	pauseMenu	Pause Menu.
	MenuView	statsMenu	Stats Menu.
	int	factorlevel	Factor to increase level. This value adjust the

			difficulty of game.
	int	level	Player leve.
	int	levelUp	This variable is used to calculate the player 's level.
PlayerController.cs	***int***	***collectedObjects***	***Collected objects used to calculate the player level.***
	int	**coins**	**Number of collected coins.**
	int	***totalCollectedObjects***	***All objects collected by player, used for stats.***
Mover.cs	Camera	cam	Main camera.
	Vector2	***startWait***	***Range of values of time wait to start the ball curve.Play with the values.***
	Vector2	***ballCurveTime***	***Range of values of duration of reverse curve.***
	Vector2	***ballCurveWait***	***Range of values of duration of reverse curve.***
	float	***spin***	***Spin of ball.***
	float	***startForce***	***Start of range force. The force will be applied to the balloon***
	float	***endForce***	***End of range force. The force will be applied to the balloon***
goalController.cs	***int***	***goals***	***Scored goals.***
GoalModel.cs	***int***	***goals***	***Number of scored goals.***
	goalController	***ComponentBehaviour***	***Access to goalController script.***
	GameObject	***gameObject***	***GameObject.***
	Transform	***transform***	***transform.***
	Vector3	***position***	***Position of***

			GameObject.
PlayerModel.cs	***int***	***savedGoals***	***total number of caught objects.***
	goalController	***ComponentBehaviour***	***Access to goalController script.***
	GameObject	***gameObject***	***GameObject.***
	Transform	***transform***	***transform.***
	Vector3	***position***	***Position of GameObject.***
	int	***coins***	***Number of caught coins.***
	int	***collectedBalls***	***Number of caught objects.***

You can play with the gravity of the prefab objects to increase the difficulty of game. The black variables are the variables that you should modify to obtain a new flow of game.

The formula below indicates the objects that player needs collect to reach a new level. For example if the factroLevel is 4 and level is 1 then (2+(1-1))*4= 8 objects to promote.

Level Formula: **(2+(LEVEL-1))*factorLevel**

1.4 Social Buttons

The social buttons perform the function of sharing the player's score on your favorite social network. The Twitter button share a tweet with the hashtags of the game and dose not have previous requirements, in the other hand, Facebook requires a developer account and app id. Both buttons opens a new windows on the browser. **NOT OPENS TWITTER OR FACEBOOK APP. TESTED ON ANDROID / PC / MAC & WEBPLAYER.**

1.4.1 Twitter button

C# Slice:

```
string twittershare=" http://twitter.com/home?
                         status="+System.Uri.EscapeDataString("#hashtag_N
#hashtag_N+1")                      +System.Uri.EscapeUriString("MYGAME new
score"+score);
Application.OpenURL(twittershare);
```

Why hashtags by separated? Because I have had troubles with url encode if hashtag was merged with message.

1.4.2 Facebook button

Right now the best solution to solve the problem of share button with facebook is use the new dialog system. The old Sharer (sharer.php) has suffered some changes.

Dialog System Example:

http://www.facebook.com/dialog/feed?
app_id=123050457758183&
link=http://developers.facebook.com/docs/reference/dialogs/& picture=http://looneybits.com/assets/img/p04.png&
name=Facebook%20Dialogs&
caption=Reference%20Documentation&description=Dialogs%20provide%20a%20simple,%20consistent %20interface%20for%20applications%20to%20interact%20with%20users.&message=Facebook%20Dialogs%20are %20so%20easy!&redirect_uri= `http:`***//www.example.com/response***

Requirements:

Facebook Account.

Facebook developer account(profile).

APP_ID. Register a simple app on ***`http://developers.facebook.com`*****.**

Hosting/server or whatever you want on the net, you need that to set the redirect_uri and pictures of game.

OLD Sharer.php System Example(Only works url data):

http://www.facebook.com/sharer.php?s=100
&p[title]=TITLE
&p[url]=http://looneybits.com
&p[summary]=yoursummaryhere
&p[images][0]=`http://looneybits.com/assets/img/p04.png`*";*

If you want to use this system you should develop a script on the server side to generate a dynamic simple score page. For example: `http://mysupergame.com/score/2000`

C# Slice

```
String facebookshare="  http://www.facebook.com/sharer.php?s=100
                        &p[title]=TITLE
                        &p[url]=http://mysupergame.com/score/" + SCORE
                        "&p[summary]=yoursummaryhere
                        &p[images][0]=http://looneybits.com/assets/img/p04.png";
Application.OpenURL(facebookshare);
```

Requirements:

Hosting/server or whatever you want on the net, you need that to set the redirect_uri and pictures of game.

1.5. Layers

The assets are distributed in several layers listed below:

- Foreground
- Background
- default

1.6. M.V.C

The project is programmed with the MVC (Model-View-Controller) architectural pattern. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

- **Model:** notifies its associated views and controllers when there has been a change in its state. This notification allows the views to produce updated output, and the controllers to change the available set of commands. In some cases an MVC implementation
- **View:** Requests information from the model that it uses to generate an output representation to the user.
- **Controller(Behavior):** can send commands to the model to update the model's state. It can also send commands to its associated view to change the view's presentation of the model

1.7.Literature

http://forum.unity3d.com/threads/is-it-really-that-hard-to-share-a-simple-score-on-facebook-and-twitter-natively.231390/ (learn about social buttons Trilusion member comment. Thanks)

https://www.assetstore.unity3d.com/en/#!/content/13866 (learn about boundary control. Thanks)

http://www.youtube.com/watch?v=N_U7GNchLZc (learn about collisions. Thanks)

https://www.assetstore.unity3d.com/en/#!/content/11228 (learn about spawn system. Thanks)

https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller (Theoretical concepts)