Jelly Cube - Code Documentation

Main Prefabs

_Game (_Game.prefab)
- GameManager.cs
    - *NextLevelName (string)*
    - *Animator (AnimatorComponent)*
- InputManager.cs
- CubeManager.cs

_UI (_UI.prefab)
- Animator (Unity component)

Cube (CubePlayer.prefab/CubeBlue.prefab/CubeRed.prefab)
- CubeController.cs

CheckPoint (CheckPointGreen.prefab, CheckPointBlue.prefab, CheckPointRed.prefab)
- CheckPoint.cs
    - *CubeTag (string)*

Camera (optional controller)
- CameraController.cs
    - LookAtTransform (Transform)
    - LookAtDamping (float)
    - CameraMoveDamping (float)

Program Flow

CubeController.cs
When the level starts, this controller register itself into a List<CubeController> inside the CubeManager.cs

InputManager.cs
When a valid touch or keyboard event is detected, it sends the roll command to the CubeManager.cs

CubeManager.cs
When there is no cubes moving, this controller can receive a roll command from the InputManager.cs. It iterates with a List<CubeController> checking wich cubes can roll. The cubes that are allowed to roll, are registered into a second list of cubes moving, and after the completion of all moves, this list is cleared. If there is no element in the list, means it´s ready to receive another roll input from the InputManager.cs. Once the cubes have finished moving, a command is sent to GameManager.cs, to check if the level is completed.

GameManager.cs
GameManager.cs controls the level loading, level completion, and ui. When it receives a command from CubeManager.cs to check if the level is completed, it will look over all checkpoints in scene to verify if they are completed.

CheckPoint.cs
When the level starts, this controller is registered into an array inside the GameManager.cs. This controller checks for any collision with a cube that has the Tag equals to its field "CubeTag". When the trigger is positive (the tags are the same), then this checkpoint is complete. A command is sent to the GameManager.cs to verify if all CheckPoints.cs in scene are completed. If the answer is yes, than the level is completed. The GameManager.cs can play the UI animation.

Main Components

GameManager.cs

_Game.prefab

Fields	Type	Description
m_NextLevelName	public string	Name of the level to be loaded when this level is complete
m_Animator	public Animator	Animator component with the animated UI
m_AnimationClipsLength	Dictionary<string,float>	Dictionary of all animations found in the m_Animator, with its respective duration
m_Checkpoints	CheckPoint[ ]	Array of all scene checkpoints
m_CheckGame	bool	Flag to enable check game completion after all cube moves
m_CameraController	CameraController	CameraController reference
Instance	GameManager	Singleton instance reference

Method	Description
CheckLevelCompletion	if m_CheckGame is true, then this method is called after all cube moves to check if the game is finished
CheckGame	Set m_CheckGame to true
ReloadLevel	Starts the ReloadLevelRoutine
ReloadLevelRoutine	coroutine that controls the UI animations and reload level
LevelComplete	coroutine that controls the UI animations and advance to next level

InputManager.cs

_Game.prefab

Fields	Type	Description
m_LockControls	bool	Ignore input events when true
m_MoveDirection	Vector2	Touch or Keyboard direction
m_Moved	bool	If moved and still touching is true
m_TouchHoldTimer	float	Hold time duration
Instance	InputManager	Singleton instance reference
TOUCH_HOLD_TIME_LIMIT	const float	Limit to start moving continuously
TOUCH_SENSIBILITY	const float	Touch sensibility to roll a cube

Method	Description
LockControls	Disable input controls
UnlockControls	Enable input controls
LateUpdate	Where all input events are calculated

CubeManager.cs

_Game.prefab

Fields	Type	Description
m_JellyEffectEnabled	public bool	Enable/Disable jelly effect
m_CubeControllers	List<CubeController>	All CubeControllers in scene
m_CubeMoving	List<CubeController>	All moving CubeControllers in scene
Instance	CubeManager	Singleton instance reference

Method	Description
Register	Add a CubeController into m_CubeControllers list
RegisterMove	Add a moving CubeController into m_CubeMoving list
UnregisterMove	Remove a CubeController from m_CubeMoving list
RollCube	Try to roll all CubeControllers registered in m_CubeControllers

CubeController.cs

CubePlayer.prefab, CubeBlue.prefab, CubeRed.prefab, CubeWall.prefab

Fields	Type	Description
m_Cube	public Collider	main cube object collider
m_CanMove	public bool	Cube can be pushed by another cube
m_CanPush	public bool	Cube can push another cube
m_CanRoll	public bool	Cube can roll after Input event
m_CanShake	public bool	Cube shakes after collision
m_RollSpeed	public float	Roll speed
m_MoveSpeed	public float	Move speed
m_Trails	public Transform	Decal prefab for trails
m_Splashs	public Transform	Decal prefab for splashs
m_LastMove	Vector3	Last move position
m_LastDir	Vector3	Last direction
SHAKE_SCALE	const float	Shake scale intensity

Method	Description
DoRoll	if m_CanRoll is true, add a tweener.cs component to roll the cube
DoPush	if m_CanPush is true, when this cube hits another cube, try to push this cube
DoMove	if m_CanMove is true, when this cube is hitted by another cube, it will move
DoShake	if m_CanShake is true, when this cube is hitted by another cube, and there is no space to move or m_CanMove is false, this cube will shake
Complete	Called after any movement completion
ResetPosition	Make sure the cube is on a flat grid rotation and position
RoundVector	Round a Vector and span it to an angle value or grid position
CreateTrail	Instantiate a trail decal when the cube moves
CreateSplash	Instantiate a splash decal when the cube is hitted by another cube

CheckPoint.cs

CheckPointGreen.prefab, CheckPointBlue.prefab, CheckPointRed.prefab

Fields	Type	Description
m_CubeTag	public string	cube tag necessary to make the checkpoint complete
m_Success	public bool	Becomes true when a cube with the tag equals to m_CubeTag field is over the object

Method	Description
OnTriggerEnter	When a cube is over the object, check if is valid
OnTriggerExit	when there is no cube over the object, set m_Sucess to false

Helper Components

CameraController.cs

CameraController.cs is a MonoBehaviour that when attached to a camera, make it look smoothly to selected transform target.

Tweener.cs

Twenner.cs is a small and very simple twenner system developed to this game. It can be replaced for any twenner you like.

JellyDecal.cs

JellyDecal.cs is a simple solution to fade and autodestroy the decals generated in runtime.

Sound.cs

Sound.cs is a basic MonoBehaviour component with an AudioSource control and a command “DontDestroyOnLoad” to make this object always active until the end of the game.

RubberEffect.cs

RubberEffect.cs is a script that changes the vertex of an object in runtime and apply a rubber behaviour on it. This is a modified version of the previous script that worked using vertex colors (http://rodrigopegorari.com/blog/?p=58). This version was modified to be rigid on bottom of the object, and soft on upper side, like it should do in real life.