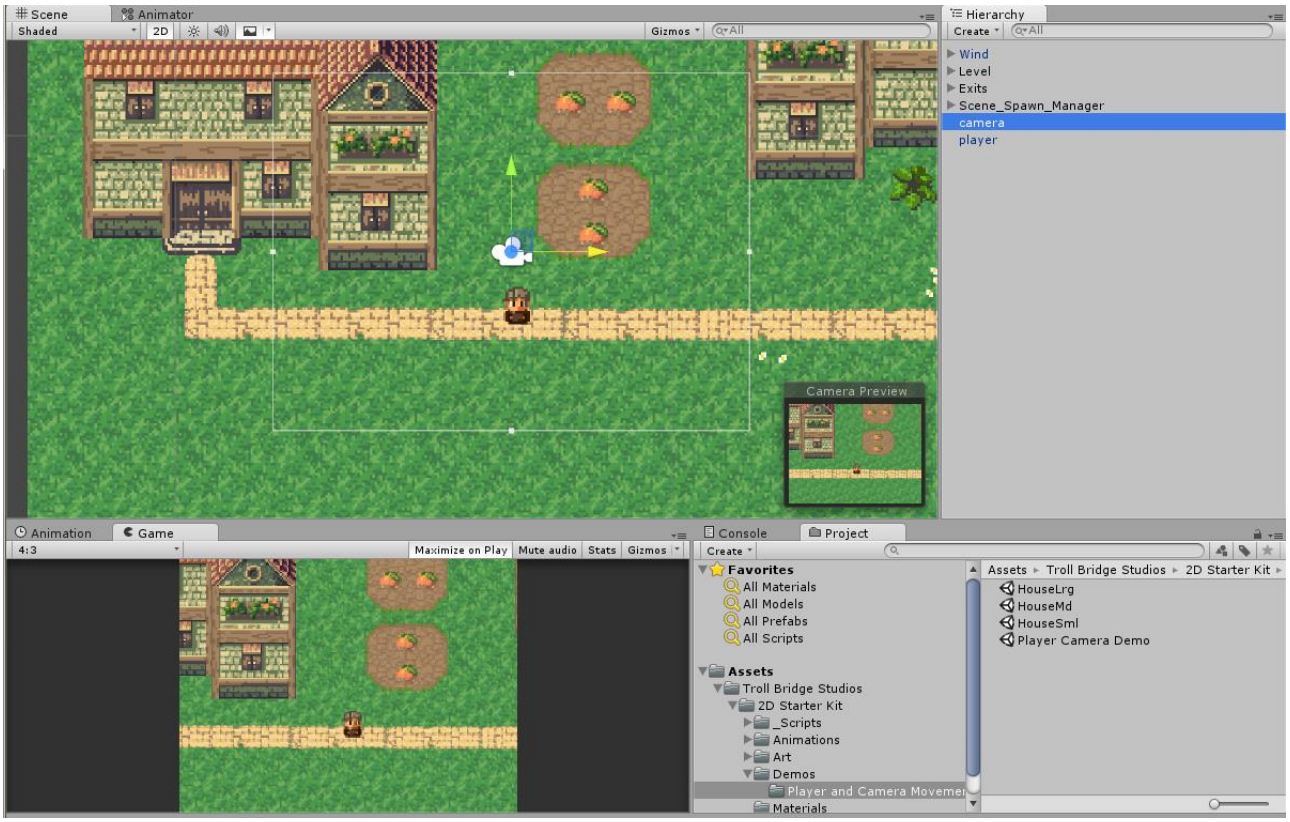
Top Down 2D RPG Kit Manual

Starter Version 1.20



February 14, 2016

Table of Contents

Game Fact #1:
In 1990 SNK created Crystalis, an action RPG that was similar to Legend of Zelda. The main difference was Crystalis allowed for eight directional movements and had a greater emphasis on combat rather than puzzle solving.

Introduction

Thank you for purchasing the Troll Bridge Studios Top Down 2D RPG Kit – Starter Version! Being a fan of the original top down RPG games like Legend of Zelda and Crystalis, I wanted to create an engine that allowed Unity developers to create games similar to these. The Starter Version focuses on setting up the core environment, player movements, and camera settings in order to help you bring your awesome game to life! Troll Bridge Studios welcomes all feedback. If you have questions, comments, or feature requests, please send an email to support@trollbridgestudios.com. If you are requesting support for this Unity Asset, please include your **Invoice Number**.

Required Tags

The following tags are required to be placed on the appropriate *GameObjects* in order to use the TBS scripts:

Required Tags		
Name	**GameObject**	**Description**
Player	Player	Required to determine what *GameObject* is the player.
Feet	Feet	Required to determine where the walking effect is assigned on the feet *GameObject*. Place this on your Player gameobject.
MainCamera	Camera	Required to determine which camera on the scene is the main camera.

Player

The player on your scene must have the following scripts in order to utilize the TBS player movement system:

- *Animator (Optional)**
- *A 2D Collider (Optional)**
- *Rigidbody 2D*
- ***Player_Manager script***

*Although the Animator and the 2D Collider are optional, they are highly recommended. The RPG Kit comes with a basic animation tree that can be utilized for simple movements, and any environment scripts require the A 2D Collider component in order to interact with the player properly.

Player Movement

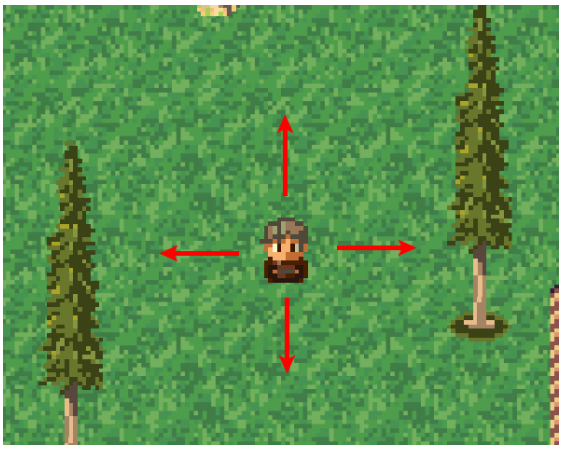
The player movement scripts are broken up into three types. Each movement type is exclusive from one another and only one movement script may be used at a time on the player. Movement scripts may be added on the player by selecting the player in the scene and either dragging the desired movement script into the player's inspector, or adding it using the *Add Component* button.

The ***Player_Manager*** script and the ***Direction_Movement*** scripts have the following public variables that can be modified to customize your player's movements.

Player Manager		
Name	**Type**	**Description**
interactionKey	KeyCode	Sets the interaction key button for interacting with objects and NPCs.
Direction_Movement Scripts		
Name	**Type**	**Description**
Speed	Float	The speed at which the player moves.

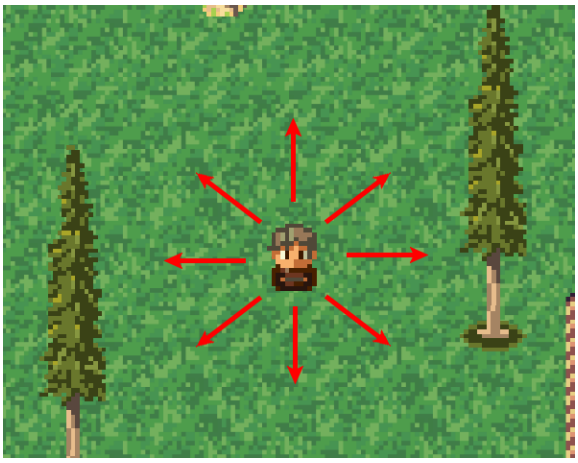
Four Directional

The ***Four_Direction_Movement*** script limits the player to only moving along the x-axis and y-axis exclusively. Attempting to move the player diagonally will have no effect.



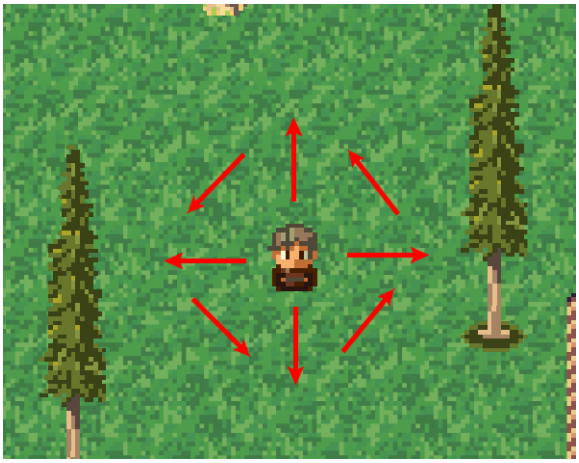
Eight Directional

The ***Eight_Direction_Movement*** script allows for the player to move along the x-axis and y-axis in 45 degree angles.



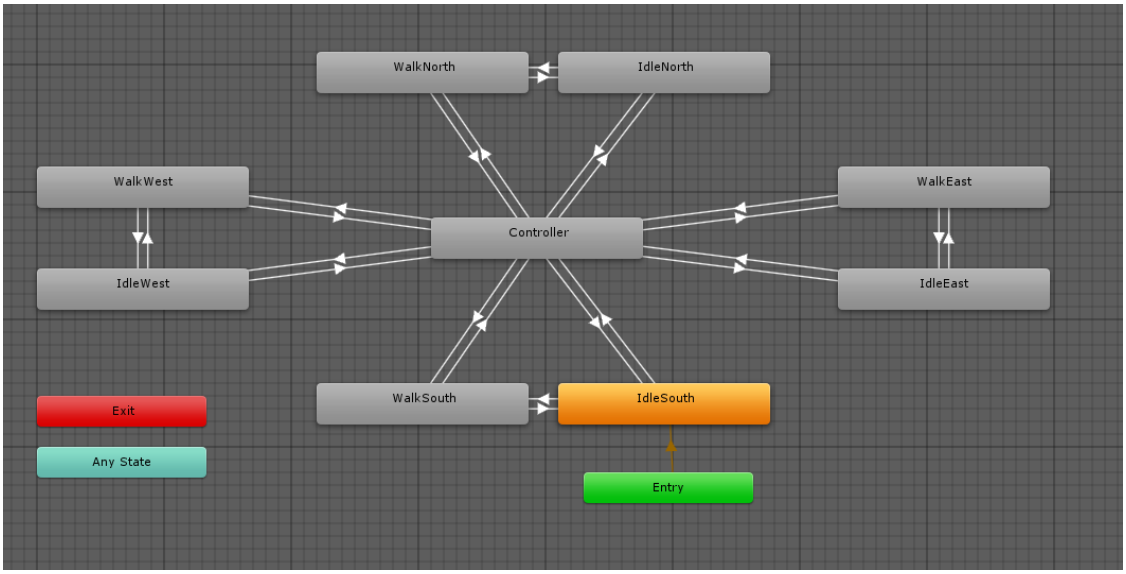
Multi Directional

The ***Multi_Direction_Movement*** script allows for the most movement out of the three types of movement scripts provided. The player may move along the x-axis and y-axis without any restrictions to moving along both axis at the same time. This movement does not start out at max speed and accelerates as the key is pushed to a max.

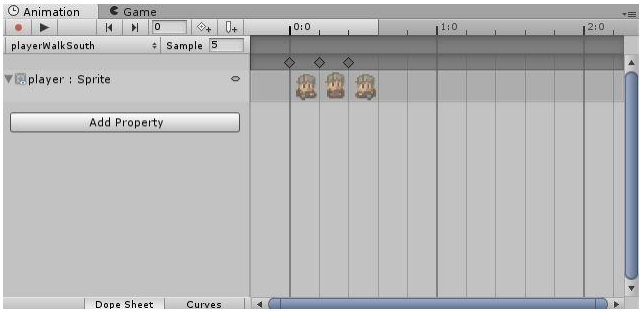


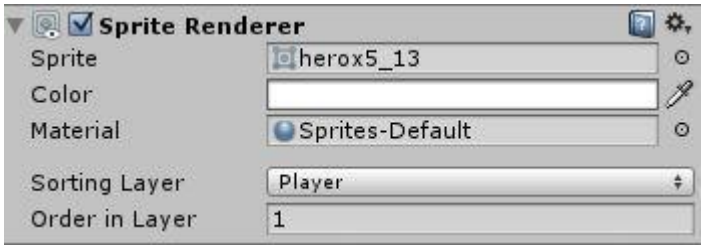
Player Animations

The default player animation tree is setup to provide animations for four facing movements: North, South, East, and West. The animations correlate with the direction the player **is moving** and in which **direction** the player is currently facing.



The animations may be replaced with your own player art and sprite sheet by replacing the *Sprite* in the *Sprite Renderer* on the player in the inspector and choosing the desired animation from within the *Animation* window. The player animator is referenced in the player movement scripts.





Game Fact #2:
Common camera resolutions today are 640 x 480 (4:3) and 1280 x 720 (16:9). The original Legend of Zelda had rooms consisting of 16 x 11 tiles (256 x 176). Each tile was 16 x 16 pixels in size and the bottom row of tiles only showed the top half of each tile.

Camera Controllers

There are three camera controller scripts available; the ***Camera_Follow_Player*,** ***Camera_Static_Slide*,** and ***Camera_Basic*** scripts. The scripts can be attached directly to the scene *Camera* object. Only one script may be attached to a *Camera* object at a time. The camera scripts may be added on the camera object by selecting the camera on the scene and either dragging the desired camera script into the camera inspector, or adding it using the *Add*

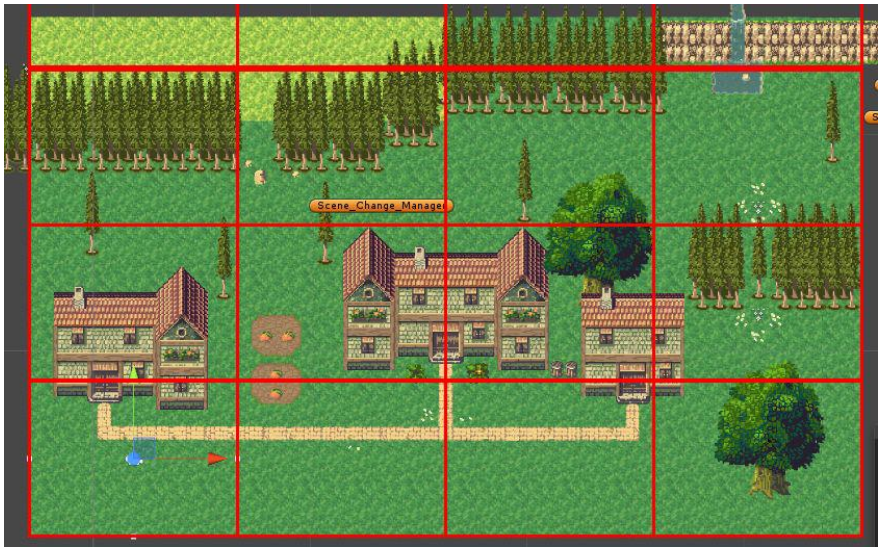
Component button. After adding the camera to the scene, make sure to set the *Tag* in the inspector to *MainCamera*.

Follow Player Camera

The follow player camera will follow the Player wherever they move with customizable restrictions based on camera borders. The **Scene Boundaries** are required *GameObjects* that tell the script where the edges of the scenes are located. The camera will not go beyond these points in the scene.

Slide Camera

The static slide camera is a room based camera that will only move when the player moves past the borders of the room. The room size is required to be set on the script in the **Camera Width** and **Camera Height** public variables. If using the static slide camera, it would be beneficially to preemptively decide the size of a standard room on the scene when building your levels. Pictured below is the demo scene split up into several rooms using a camera width and height of 1024x704, starting the camera in the bottom left corner of the scene.



Basic Camera

The main menu camera can be used as a non-moving camera on scene with a cut scene or a main menu. The camera has simple functionality and is used to display a scene without the player.

The ***Camera_Follow_Player***, ***Camera_Static_Slide***, and ***Camera_Basic*** scripts have the following public variables that can be modified to customize how the cameras operate on the scene.

Camera_Follow_Player		
Name	**Type**	**Description**
Camera Width	Float	The Camera Width.
Camera Height	Float	The Camera Height.
Is Camera Moving	Boolean	The Is Camera Moving checkbox can be checked if the camera is not meant to move no matter where the player moves on the scene. By default this is set to false.

Bottom Camera Border	GameObject	The bottom edge of the scene. The camera will not go further south of this GameObject.
Left Camera Border	GameObject	The left edge of the scene. The camera will not go further West of this GameObject.
Top Camera Border	GameObject	The top edge of the scene. The camera will not go further North of this GameObject. If you have a UI that is not transparent then you will need to set a GameObject for this that will compensate for the UI height so the UI doesn't interfere with your scene.
Right Camera Border	GameObject	The right edge of the scene. The camera will not go further East of this GameObject.
Camera_Static_Slide		
Name	**Type**	**Description**
Player Slide Amount	Float	The amount to move the player into a new room when panning the camera to a new room.
Camera Width	Float	The Camera (Room) Width.
Camera Height	Float	The Camera (Room) Height. If you have a UI that is not transparent then you will need to incorporate the height of the UI into the total height.
Camera Slide Speed	Float	How fast the camera will pan to a new room.
Bottom Boundary	GameObject	Setting the bottom boundary will automatically adjust your camera position.
Left Boundary	GameObject	Setting the left boundary will automatically adjust your camera position.
Camera_Basic		
Name	**Type**	**Description**
Camera Width	Float	The Camera width.
Camera Height	Float	The Camera Height

Environment

TBS provides different ways to create an interactive environment. Currently, TBS has Pushable Terrain, An Effectors system (Unity 5.0), Teleportation, Destroy Triggers, Terrain Sounds, Area and Action Key Dialogue, and Icon Displays.

Creating Pushable Terrain

The ***Terrain_Pushable*** script allows GameObjects to be pushed by another object. The **Rigidbody 2D** and **Collider 2D** components are required to use the script. The following public variables can be used to customize how the object is pushed.

Terrain_Pushable		
Name	**Type**	**Description**
Movable	Boolean	Determine whether the object is allowed to be moved.
Layers That Move It	LayerMask	Determines which layers can move this object. For example, if you want the player to move the object, choose the player layer you've assigned to the player GameObject.
Time To Push	Float	The delay time it takes to push the object.
Move Speed	Float	The speed at which the object gets moved. Range [0-5].
Up, Down, Left, Right	Transform	Where to push this object on all four sides. A minimum of one transform must be assigned to the direction for the Pushable script to work. Assigning these variables allows the Pushable object to be customized allowable movement directions.
Show Rays	Boolean	Show the Raycast in the Scene View while the game is playing. This can be used to help test and debug.
Sound Clip	AudioClip	The sound clip to play when the object is being moved / pushed.
Minimum Pitch	Float	The minimum pitch this sound can be played at. A random

		number between the minPitch and a maxPitch will be chosen.
Maximum Pitch	Float	The maximum pitch this sound can be played at. A random number between the minPitch and a maxPitch will be chosen.

Effectors System

The TBS 2D engine utilizes the Unity Effectors that were introduced in Unity 5.0. The Unity Effectors are a set of 2D physics effectors that can be used to simulate effects that interact with the player like wind, sand, and water. There are two examples on the demo that utilize the **Area Effector Box** prefab. The first is a **Wind** prefab; this uses the Area Effector box with a particle system to simulate debris. The second is an Area Effector Box that has the ***Terrain_Collision_Animation*** script attached. This script allows for an animation to be played when the effector is colliding with an object. The following variables are used to customize the collision animation.

Terrain_Collision_Animation		
Name	**Type**	**Description**
Terrain Animation	GameObject	The animation to be played while colliding.

Teleportation

The teleportation system allows the player to teleport between two locations on the same scene. The teleportation system can be used by adding the script to any object with an *Is Trigger Collider 2D.* It is recommended to attach the ***End_Of_Animation_Destroy*** script to the animation GameObject, placing the .anim in the objectAnimation public variable.

Target_Teleport		
Name	**Type**	**Description**
NewLocation	Transform	Position the tagged object(s) will appear after colliding with the teleportation collider.
Specific Tag(s) for teleport	Array	The tag of the object(s) that will be teleported if it collides with the teleportation collider. The demo scene provided uses the **Player** tag, allowing the player object to teleport when they collide with the circle collider.
Teleport Start Animation	PreFab	Teleport animation for starting destination.
Teleport End Animation	PreFab	Teleport animation for ending destination.
Sound Clip	AudioClip	The sound clip to play when teleporting.
Minimum Pitch	Float	The minimum pitch this sound can be played at. A random number between the minPitch and a maxPitch will be chosen.
Maximum Pitch	Float	The maximum pitch this sound can be played at. A random number between the minPitch and a maxPitch will be chosen.
End_Of_Animation_Destroy		
Name	**Type**	**Description**
objectAnimation	AnimationClip	The .anim file that will be destroyed when the animation has reached the end.

Destroy Triggers

The destroyable system allows for triggers to be setup on different areas of the scene. These triggers can be used to achieve an interactive environment where passing through a *Collider 2D trigger* will destroy an object. These triggers are meant to be used to create things like fake walls and puzzles. For example, on the demo scene, moving the bookshelf in the large house destroys the wall hiding a chest. There are three different methods and correlating scripts to achieve this behavior.

- ***Destroy_By_Colliding_By_Parent***: Selecting a GameObject for the parent will destroy all the children of this gameobject when activated. *Example: Destroying a room full of enemies that are a part of a parent GameObject.*
- ***Destroy_By_Colliding_By_Tag***: Assigning a tag will destroy all GameObjects with this tag when activated. Example: *Destroy only GameObjects with tag "Slimes."*
- ***Destroy_By_Colliding_Manually***: Assigned GameObjects will be destroyed when activated.

The following public variables are used to customize the destroy scripts.

Destroy_By_Colliding_By_Tag		
Name	**Type**	**Description**
Set Inactive	Boolean	Set to 'True' if you want to set the gameobjects inactive rather than destroying.
On Enter Collision	Boolean	Do we destroy when we first enter collision?
Trigger Amount	Float	The amount of times this can be triggered. Assigning this to -1 will make it run infinite.
Tags that Activate	Array	The tags that can only activate the Destroy.
Tags To Be Destroyed	Array	The GameObject(s) with these tags will be destroyed.
On Exit Collision	Boolean	Do we destroy when we first exit collision?
Trigger Amount	Float	The amount of times this can be triggered. Assigning this to -1 will make it run infinite.
Tags that Activate	Array	The tags that can only activate the Destroy.
Tags To Be Destroyed	Array	The GameObject(s) with these tags will be destroyed.
Destroy_By_Colliding_By_Parent		
Name	**Type**	**Description**
Set Inactive	Boolean	Set to 'True' if you want to set the gameobjects inactive rather than destroying.
On Enter Collision	Boolean	Do we destroy when we first enter collision?
Trigger Amount	Float	The amount of times this can be triggered. Assigning this to -1 will make it run infinite.
Tags that Activate	Array	The tags that can only activate the Destroy.
Destroy these Parents Children	Array	These parent GameObject(s) will have all their children destroyed.
On Exit Collision	Boolean	Do we destroy when we first exit collision?
Trigger Amount	Float	The amount of times this can be triggered. Assigning this to -1 will make it run infinite.
Tags that Activate	Array	The tags that can only activate the Destroy.
Destroy these	Array	These parent GameObject(s) will have all their children

Parents Children		destroyed.
Destroy_By_Colliding_By_Manually		
Name	**Type**	**Description**
Set Inactive	Boolean	Set to 'True' if you want to set the gameobjects inactive rather than destroying.
On Enter Collision	Boolean	Do we destroy when we first enter collision?
Tags that Activate	Array	The tags that can only activate the Destroy.
GameObjects to Be Destroyed	Array	The GameObjects that are manually placed in here will be destroyed.

Terrain Sounds

The terrain sounds system is used to play sound FX when the player walks across certain terrain. For example, in the Player and Camera Movement demo, when the player walks into one of the houses, wood footsteps will be played as the player moves through the house. The ***Terrain_Sound_By_Distance*** can be added to a *gameobject* with a 2D *Collider 2D trigger*. The following public variables are used to customize the terrain sound system.

Terrain_Sound_By_Distance		
Name	**Type**	**Description**
SoundClip	AudioClip	The sound clip to play when Colliding.
minPitch	Float	The min pitch for when this sound is played.
maxPitch	Float	The max pitch for when this sound is played.
distance	Float	The distance before playing another sound.

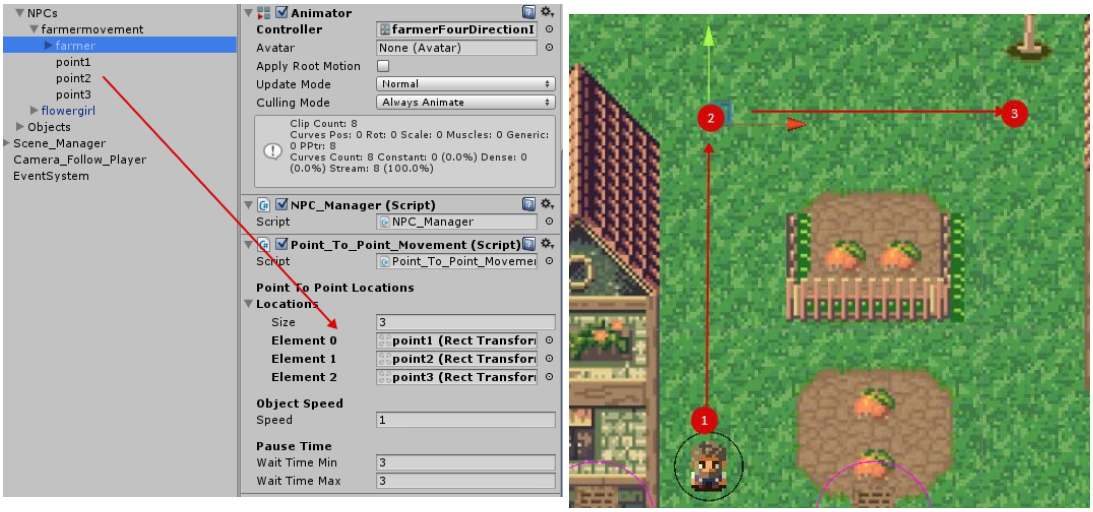
NPC Scripts

There are several scripts can be used for basic NPC interactions and movement. The ***Point_To_Point_Movement*** script can be used to have an NPC roam between arrays of points, utilizing the four directional movement animator to animate the NPC as it moves between these points. The ***Action_Key_Dialogue*** and ***Area_Dialogue*** can be used for NPC interaction, while the ***Icon_Display*** can be used to create objects that float above the NPC's head, such as an exclamation mark to indicate a quest giver. These scripts can be used in a variety of ways. For example, both dialogue scripts can be used for signs, while the icon script can be used to give the player a variety of feedback on the game world (i.e. chests, pushable objects, clues).

Gameobjects on your scene must have the *Rect Transform* in order to utilize the Dialogue and Icon systems properly, and be attached to a *Canvas* with *World Space selected* as the *Render Mode*.

Point to Point Movement

The movement script uses an array of gameobjects to create an effect of a roaming NPC, bringing your NPCs to life. The ***Point_To_Point_Movement*** script must be attached to NPC *GameObject* and points defined and attached via an array.

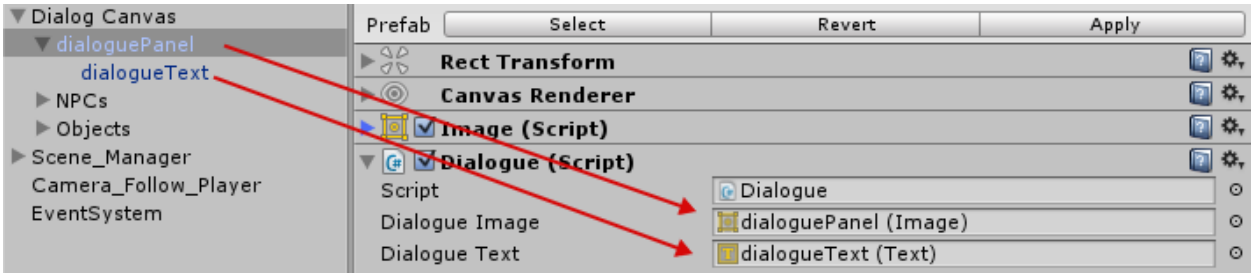


The following public variables are used to customize the movement system.

Point_To_Point_Movement		
Name	**Type**	**Description**
Locations	Array	The order in which the *GameObject* will be moving towards, once the end is reached it will traverse backwards through the array.
Speed	Float	The speed at which the *GameObject* will be moving.
Wait Time Min	Float	The minimum time the *GameObject* waits before going to the next 'Location.'
Wait Time Max	Float	The maximum time the *GameObject* waits before going to the next 'Location.'

Dialogue Scripts

The dialogue scripts can be used to make any *GameObject* have a certain amount of interactivity. *Gameobjects* on your scene must have the *Rect Transform* in order to utilize the Dialogue and Icon systems properly, and be attached to a *Canvas* with *World Space* selected as the *Render Mode*. In order to use the TBS dialogue scripts, the ***Dialogue*** script is required to be attached to the Dialogue Panel UI preFab. This prefab is then set on the ***Action_Key_Dialogue*** or ***Area_Dialogue*** in the *Dialogue UI* public variable.



The following public variables are used to customize the dialogue system.

Action_Key_Dialogue		
Name	**Type**	**Description**
Show Area in Scene	Boolean	Set to true if you want to see the NPC area that the Player can interact in the scene.
Area Collider	Collider2D	The Collider2D that represents the range for the Interaction to happen.
Area Color	Color	Sets the color of the *Collider 2D* that is being shown in your scene view.
Prevent Area Dialogues	Boolean	Set to 'True' if you want all Area Dialogues that are currently showing and Area Dialogues that could show during this dialogue to not be shown.
Freeze Player	Boolean	Set to 'True' if you want the player to not be able to move while the dialogue is up and running.
Dialogue UI	GameObject	The Dialogue UI *GameObject* that will be displayed.
Dialogue Color	Color	The color alteration for the 'Dialogue UI.' Leaving this color 'white' will keep the same look for your 'Dialogue UI' *GameObject*.
Dialogue Space	Float	The distance above the *GameObject* for the dialogue's location.
Multiple Transitions	Boolean	Set to 'True' if you want dialogue UI transitions to happen after each dialogue string in the 'Dialogue' array.
Instant Transition	Boolean	Set to 'True' if you want the dialogue UI transitions to appear/disappear instantly.
Fade Transition	Boolean	Set to 'True' if you want the dialogue UI transitions to fade.
Fade Time	Float	The fade time for when a dialogue box fades in and fades out.
Grow/Shrink Transition	Boolean	Set to 'True' if you want the dialogue UI transition to grow and shrink.
Grow/Shrink Time	Float	The grow/shrink time for when a dialogue box grows in and shrinks out.
Text Color	Color	The color of the dialogue text.
Instant Text	Boolean	Set to 'True' if you want the text transition to appear/disappear instantly.
Faded Text	Boolean	Set to 'True' if you want the text transitions to be faded in and out.
(Text) Fade Time	Float	The time at which the text is faded in and out.
Typed Text	Boolean	Set to 'True' if you want the text transition to be typed out.
Pause Time	Float	The time is takes for the next letter to be displayed. Increasing this number slows the typing speed of the dialogue text while decreasing this number speeds up the typing speed of the dialogue.
Type Sound	AudioClip	(Optional) The sound that plays when each character is typed in the dialogue.
Dialogue	Array	The text that is displayed in the Dialogue UI. Each array element creates a new set of dialogue that is scrolled through with the 'Action' key.
Area_Dialogue		
Name	**Type**	**Description**
Show Area in Scene	Boolean	Set to true if you want to see the NPC area that the Player can interact in the scene.
Area Collider	Collider2D	The Collider2D that represents the range for the Interaction

		to happen.
Area Color	Color	Sets the color of the *Collider 2D* that is being shown in your scene view.
Dialogue UI	GameObject	The Dialogue UI *GameObject* that will be displayed.
Dialogue Color	Color	The color alteration for the 'Dialogue UI.' Leaving this color 'white' will keep the same look for your 'Dialogue UI' *GameObject*.
Dialogue Space	Float	The distance above the *GameObject* for the dialogue's location.
Inactive Time	Float	The amount of seconds that dictate how long before the dialogue can be displayed again after it has been completed/destroyed.
Multiple Transitions	Boolean	Set to 'True' if you want dialogue UI transitions to happen after each dialogue string in the 'Dialogue' array.
Instant Transition	Boolean	Set to 'True' if you want the dialogue UI transitions to appear/disappear instantly.
Fade Transition	Boolean	Set to 'True' if you want the dialogue UI transitions to fade.
Fade Time	Float	The fade time for when a dialogue box fades in and fades out.
Grow/Shrink Transition	Boolean	Set to 'True' if you want the dialogue UI transition to grow and shrink.
Grow/Shrink Time	Float	The grow/shrink time for when a dialogue box grows in and shrinks out.
Text Color	Color	The color of the dialogue text.
Instant Text	Boolean	Set to 'True' if you want the text transition to appear/disappear instantly.
Faded Text	Boolean	Set to 'True' if you want the text transitions to be faded in and out.
(Text) Fade Time	Float	The time at which the text is faded in and out.
Typed Text	Boolean	Set to 'True' if you want the text transition to be typed out.
Pause Time	Float	The time is takes for the next letter to be displayed. Increasing this number slows the typing speed of the dialogue text while decreasing this number speeds up the typing speed of the dialogue.
Type Sound	AudioClip	(Optional) The sound that plays when each character is typed in the dialogue.
Dialogue	Array	The text that is displayed in the Dialogue UI. Each array element creates a new set of dialogue that is scrolled through with the 'Action' key.
Dialogue		
Name	**Type**	**Description**
Dialogue Image	Panel	The panel that will be used for the Dialogue UI in the dialogue scripts.
Dialogue Text	Text	The text that will be used in the dialogue scripts.

Icon Display

The Icon_Display script can be used to display a *prefab* above the *GameObject* when the player makes contact with a *Collider 2D*. The object can be set to bounce, pulse, and spin. The following public variables are used to customize the movement system.

Icon_Display		
Name	**Type**	**Description**
Icon	GameObject	The *GameObject* to appear as the icon.

Space for Icon	Float	The distance above the *GameObject* for the icon's location.
Bounce Icon	Boolean	Set to 'True' if you want the icon's movement to bounce.
Bounce Distance	Float	The distance the icon moves before it moves back to its original location.
Bounce Time	Float	The time it takes for the icon to move a distance of 'Bounce Distance'
Pulse Icon	Boolean	Set to 'True' if you want the icon's movement to pulse.
Pulse Intensity	Float	How intense the pulse should be. IF set above 1, the pulse will go big-small-big. IF set under 1, the pulse will go small-big-small.
Pulse Time	Float	The time it takes for the icon to pulse from small-big or big-small.
Spin Icon	Boolean	Set to 'True' if you want the icon's movement to spin.
X Spin Speed	Float	The speed the *GameObject* rotates on its X.
Y Spin Speed	Float	The speed the *GameObject* rotates on its Y.

Scene Controllers

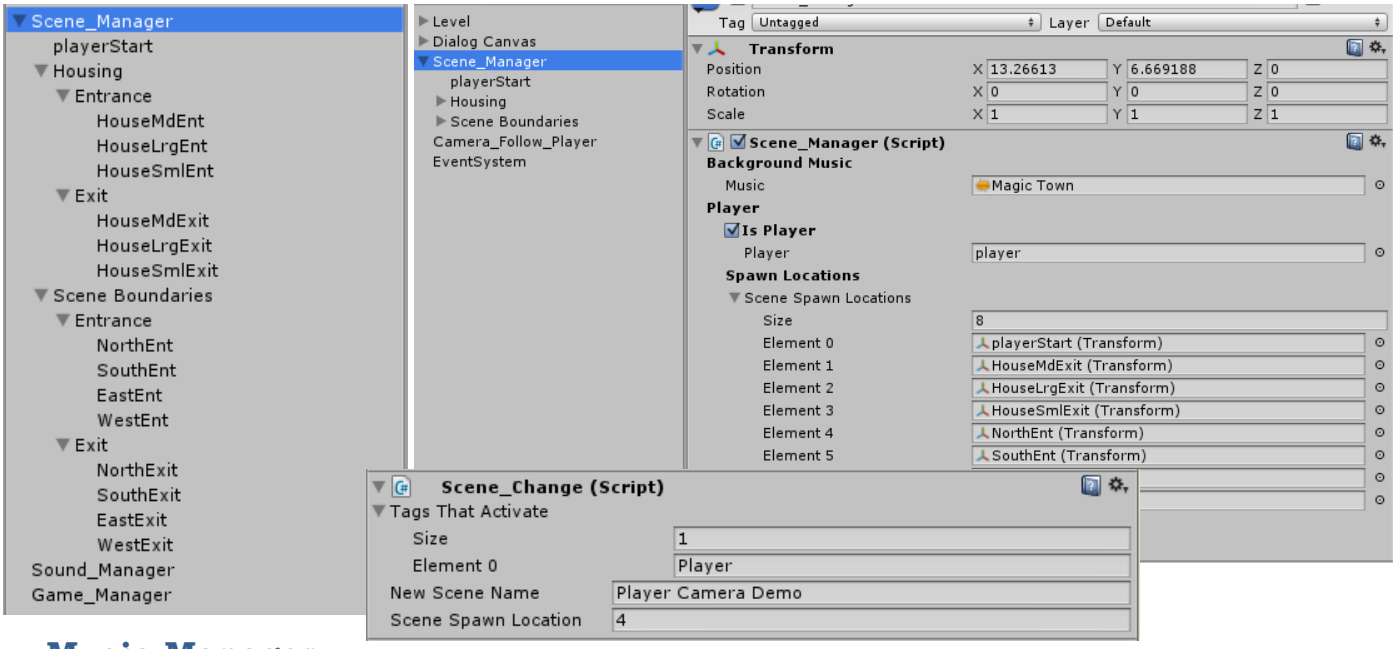
The scene controllers are used to go between scenes and manage entrance and exit points. The ***Scene_Manager*** script is used in conjunction with the ***Scene_Change*** script to determine where the player spawns when moving between scenes. An array of *Scene Spawn Locations* is defined on the ***Scene_Manager*** script.

Scene Spawn Locations

The Scene Spawn Locations system is used to handle where the player is spawned on any given scene. The player locations can be customized from scene to scene. The locations are assigned on the ***Scene_Manager*** GameObject in an array and referenced when traveling between scenes.

Scene Manager

The Scene Manager system is used to determine where the player starts on the scene, any entrances or exits that send the player to another scene, and where the player enters the scene from another one. It also contains the scene **Music**, and the **Player** *GameObject*. Please refer to the demo scene for an example of how the Scene Manager system works.



Music Manager

In the TBS Top Down 2D RPG engine the soundtrack is handled on the **Scene_Manager** object. The **SoundManager** GameObject is used to control whether the sounds are on or off, and change the volume. The **Scene_Manager** script contains a public variable called ***Music*** where an .ogg file may be assigned to be played on the scene. **Note: If you would like for the same song to continue playing when transitioning between scenes, make sure the same song is placed on both scenes.**

Sound_Manager		
Name	**Type**	**Description**
Music On	Boolean	Allows you to turn the music on or off.
Sfx On	Boolean	Allows you to turn the Sfx on or off.
Sfx Volume	FloatRange	The Sfx Volume Range [0-1].
BG Music Source	AudioSource	The background music to be assigned for manipulation.
Scene_Manager		
Name	**Type**	**Description**
Music	AudioClip	The music to be played on the scene. If you would like the music to continue from one scene to another, assign the same music onto both scenes.

User Interface

The 2D Kit UI system allows for a basic menu system, and an options menu. These can be implemented on any scene a customized based on the scene needs. For example, the *Esc* key

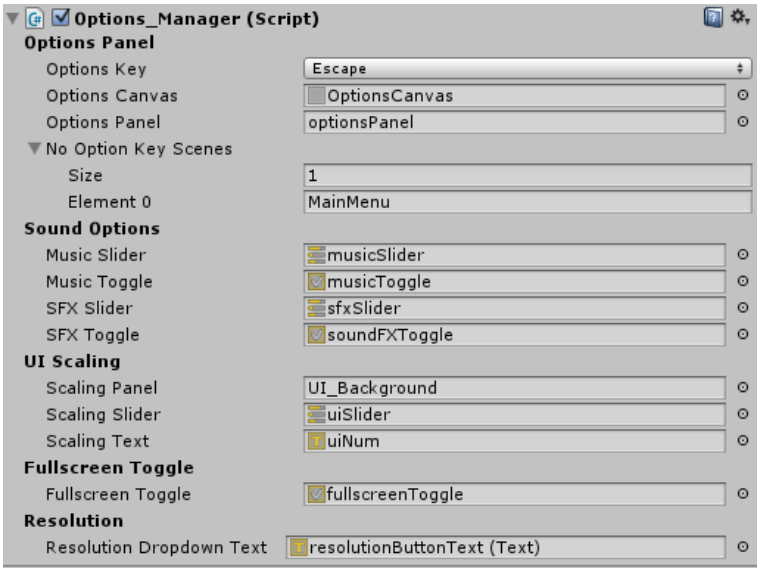
on a keyboard is commonly used to enable an options menu. However, this function would most likely be disabled on a main menu scene, as there would be an *Options* button instead. This key can not only be customized in the inspector, but disabled on specific scenes. This feature would be beneficial for a cut scene.

The options menu provides the following functionality:

- Music Volume Sliders
- Sound Volume Sliders
- Music Disable / Enable Toggle
- Sound Disable / Enable Toggle
- Scaling UI Slider
- Fullscreen Toggle

Options Manager

The ***Options_Manager*** script can be assigned to your *Canvas* on the scene. Assignments need to be made within the *Hierarchy* in order to utilize the script's functionality. Refer to the demo scene in order to understand how the scripts are assigned to each function within the ***Options_Manager*** script.



Patch Notes 1.1

Cameras
-1:1 camera resolution now functional
Camera Static
- Fixed camera panning bug, camera will now pan the correct amount
- Has 2 variables which the user puts in the bottom and left boundaries so the camera auto positions where the player spawns based on camera dimensions.
- Added Debug for the 2 new variables
NEW - Camera Basic
- Added Editor
- Debug checks added.
- Located in _scripts -> camera -> camera_basic
- Additional notes for Camera Basic -> Non moving camera with camera dimension settings.

Target Teleport
- Teleport sound added with pitch
Terrain Pushable
-Fixed a bug when playing sound

NEW - Terrain Sound By Distance
- Play sounds while an object is moving in an area specified by the user and
- Play sounds based on the distance traveled.
- Added Editor
- Debug check added
- Located in _scripts->Environment->Terrain

Helper Manager
- Public variable added for placing the UI of your game.
- More functions added
- Singleton code reworked/recoded.

NEW - Options Manager
- added editor
- debug check added
- UI Scaling added
- Fullscreen and windowed control added
- Resolution change added
- customized hotkey button to bring up the options menu
- mute/unmute toggle for SFX and music added
- volume slider for SFX and music added
- set which scenes the hotkey will bring up options panel
- located in _scripts->game helper managers -> options manager

NEW - Resolution Change
- added debug check
- changes screen resolutions based on width and height
- located in _scripts-> game helper managers ->resolution change

Scene Manager
- Added debug check
- Updated editor
- Toggle if the player is needed to be spawned in the scene
- Toggle if the UI is needed to be displayed.

Sound Manager
- Removed music slider on the script
- Singleton code reworked/recoded
NEW - Dont Destroy On Scene Load
- Keeps game object alive when a scene changes
NEW - Grab Main Camera On Canvas
- Sets canvas render camera to the camera with the tag "MainCamera"
- Located in _Scripts-> Camera -> Grab Main Camera On Canvas

Patch Notes 1.2

Cameras

-Camera follow and Camera Slide have had their tooltips adjusted to help if using a transparent UI or not.

Player

-The **Player_State** script now holds all of the player's state (i.e. CanPlayerMove, PlayerAlterSpeed, and PlayerInvertX). Changing the default values in this script will have an effect on your player. This script will be used later to manage combat and effects the player may incur, such as slow, confusion spells. This script has a direct relationship with the **Player_Manager** script.

-The 'Interaction Key' can be selected in the **Player_Manager** script. The 'Space' button is used by default in the demo.

NEW – Action Key Dialogue

- The **Action_Key_Dialogue** script can be used for NPC interaction. It allows the player to interact with NPCs with a predetermined 'Action Key' (i.e. spacebar).

NEW – Area Dialogue

- The **Area_Dialogue** script can be used for NPC interaction. It allows the player to interact with an object or NPC when coming within a certain distance.

NEW – Icon Display

-The **Icon_Display** script can be used to display a preFab above a GameObject when coming within a certain distance.

UI

-The UI_Background has been updated to use *Rect Transform* instead transform and set to '*Top Stretch*'

Scene Manager

-Scene manager has been moved to its own *GameObject*.

Support

For support, please email directly
support@trollbridgestudios.com

Please include your Asset Store Invoice ID in the email.