# SKCell

v1.2.4 by **Alex Liu**

SKCell is a powerful, comprehensive utility package for Unity that can greatly enhance your development experience.

Webpage: **here**

Latest release: **here**

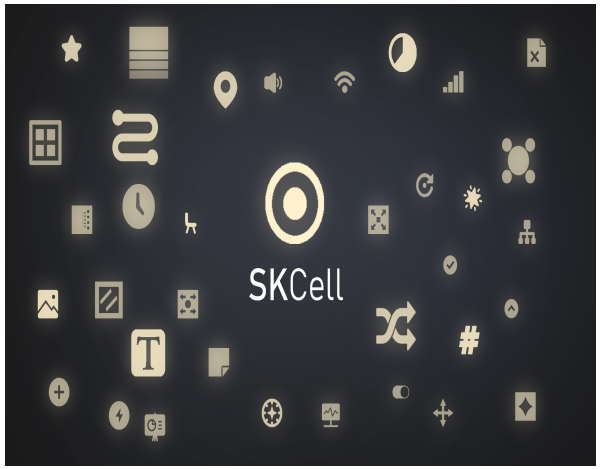
*Dev log is at the end of the document.*

## Features

**LOTS of utility functions!**

One-line tweening and timed function calls, object pools, audio and video players, FSMs, CSV spreadsheet reader, game event system, 200+ util functions and extensions, scene loader, singleton and command patterns, etc.

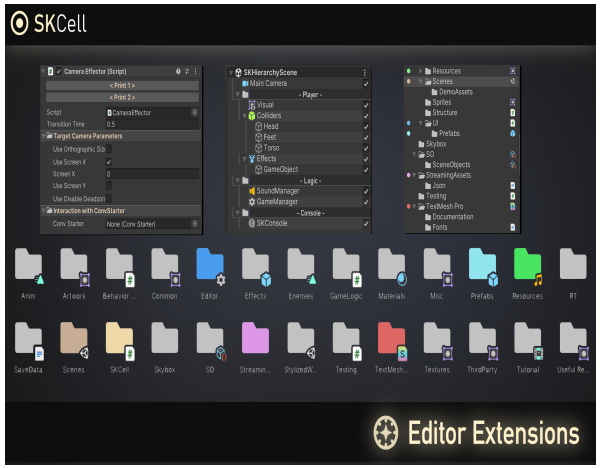
Head to **Common Utilities** for more details.



### Better Editor Windows

Improved Unity editor windows (hierarchy, transform, project) with more accessible utilities and a better feel.

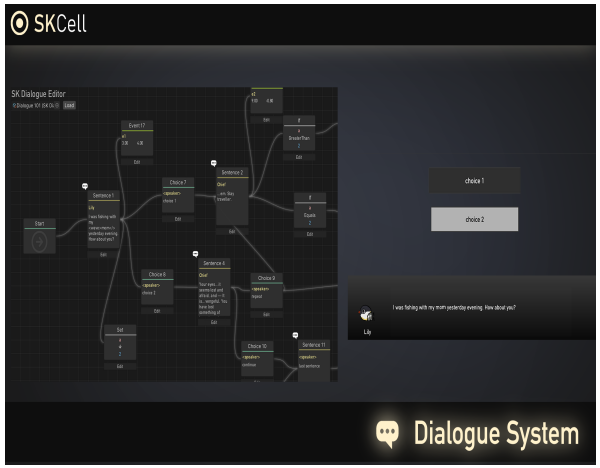
See: **Editor Windows**




### Dialogue System

Visualize your dialogue flow with a node editor. Multiple logic nodes are availale, such as options, random, boolean if/else, and custom events.

See: **Dialogue System**




### Text Animator

Quickly implement typewriter and text effects simply by labeling your texts. There are 17

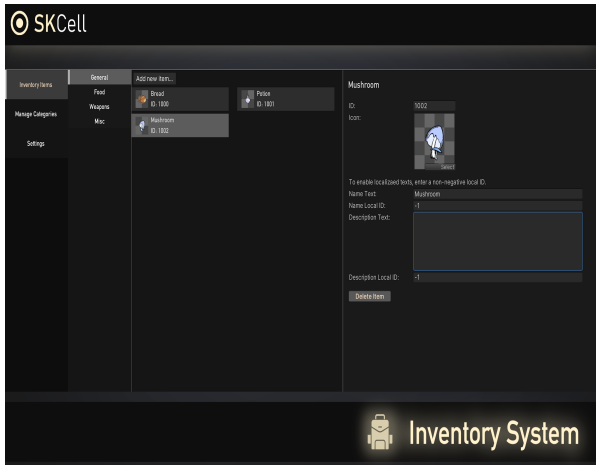
built-in presets with customizable parameters.

See: **Text Animator**


### Inventory System

Build your item database in Unity and have a fully-functional inventory UI ready in seconds.

See: **Inventory System**




### Image Effects

Customizable sprite and image effects including color grading, outline, shadow, blur, fading, and custom blend modes.

See: **Effects**


### Revamped UI Components

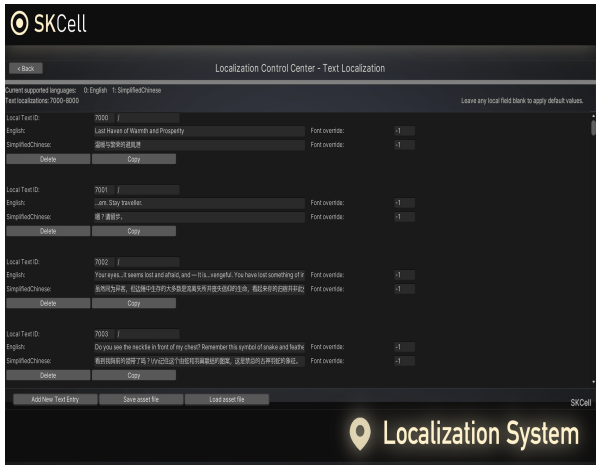
Versatile and time-saving UI components including buttons, sliders, toggles, toggle groups, panels, scrollers, texts, images, and UI particle systems.

See: **UI**


### Localization

Integrated localization system with full editor window. Automatically works with SKCell UI components.

See: **Localization**




### Grid System

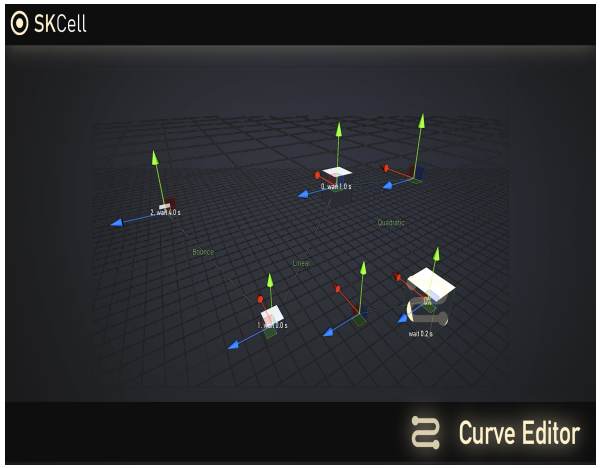
Powerful grid system with pathfinding. Grid values, costs, obstacles, and savable assets are also available.

See: **Grid System**


### Object Path Designer

Design a path for your Game Objects by setting multiple waypoints, wait times, movement tweening curves, and bezier curves.

See: **Path Designer**




### In-game Console

Build your own in-game console within seconds! SKCell offers a powerful, extendable console system.

See: **In-game Console**


## Getting Started

SKCell is compatible with **Unity 2021.x** and newer.

### Install

To integrate SKCell into your project, simply download the latest release (unity package). Published releases can be found on the right side of the github repo page. Then, import the package into your project resource folder.
Notice a folder named **SKCell** will appear under the Assets directory.

### Setup

To make SKCell functions available, create a new game object and add the **SKCore** component. This automatically adds some of the core components that SKCell relies on. Make sure **SKCommonTimer** and **SKPoolManager** are attached.

### Use

Include the **SKCell namespace** in your scripts to access SKCell classes. Try the following line of code to see if it works!

csharp

```csharp
using SKCell;
//...
SKUtils.EditorLogNormal("Hello SKCell!");
```

Congratulations! You have now set up SKCell. Please check the documentation to see what it can do for you.

# Documentation

(This documentation is up to date for v1.2.4)

# 1. Common Utilities

SKCell provides an abundency of useful utility functions, mostly in the static class **SKUtils**.

## 1.1 Logging

There are 5 logging functions (like Debug.Log) corresponding to 5 critical levels:

**void EditorLogSafe(object message, bool detailed = false)**

**void EditorLogNormal(object message, bool detailed = false)**

**void EditorLogWarning(object message, bool detailed = false)**

**void EditorLogError(object message, bool detailed = false)**

**void EditorLogCritical(object message, bool detailed = false)**

*- Logs a message to the console with a time stamp.*

*- detailed: display the time difference between current and last call.*

csharp

```
//EXAMPLE
SKUtils.EditorLogError("Error message: item ID is -1!");
```

## 1.2 Debugging

**void DebugDrawCircle(Vector3 center, float radius, Color color, float duration, int divisions)**

*- Draw a circle using the Unity Debug.DrawLines.*

*- divisions: Number of lines forming the circle. Higher divisions give smoother shapes.*

**void DebugDrawRectangle(Vector3 minXY, Vector3 maxXY, Color color, float duration)**

*- Draw a rectangle using the Unity Debug.DrawLines.*

*- minXY: x and y coordinates of the lower left corner. (z component is not used)*

*- maxXY: x and y coordinates of the upper right corner. (z component is not used)*

**void DebugDrawText(string text, Vector3 position, Color color, float size, float duration)**

*- Draw a string of text using the Unity Debug.DrawLines.*

csharp

```
//EXAMPLE
SKUtils.DebugDrawText($"Position: {pos}",pos, Color.Red, 25.0f, 10.0f);
```

## 1.3 Coroutines

**void InvokeAction(float seconds, Action callback, int repeatCount = 0, float repeatInterval = 1, string id = "", Action onFinish = null)**

*Invokes an action after **time** seconds, then repeatedly every **repeatInterval** seconds, stopping at **repeatCount** times.*

*callback: The function you want to call.*

*id: ID for this coroutine. Use it to stop the coroutine later.*

*onFinish: The function to call after this coroutine is finished.*

**void InvokeActionUnlimited(float seconds, Action callback, float repeatInterval = 1, string id = "")**

*Invokes an action after **time** seconds, then repeatedly every **repeatInterval** seconds. Will not stop.*

### void InvokeActionEditor(float seconds, Action callback)

*Invokes an action after **time** seconds in the editor.*

csharp

```
//Spawn Effect and Update UI after 3 seconds, then repeat 5 times every 1 s
SKUtils.InvokeAction(3.0f, ()=>
{
    SpawnEffect();
    UpdateUI();
},5,1);
```

Nested calls are also supported!

csharp

```
//Do something after 1 second, then do something after 2 seconds.
SKUtils.InvokeAction(1.0f, ()=>
{
    //...
    SKUtils.InvokeAction(2.0f, ()=>
    {
        //...
    });
});
```

### void CancelInvoke(string id)

*Stops an ongoing InvokeAction call specified by its id.*

csharp

```
//EXAMPLE
SKUtils.InvokeAction(3.0f, ()=>
{
  //...
},0,0,"action_07");
//...
SKUtils.CancelInvoke("action_07");
```

## 1.4 Tweening

Make animation tweening effects with just a few lines of code!

**void StartProcedure(SKCurve curve, float time, Action <float> action, Action<float> onFinish = null, string id = "")**
*Starts a continuous procedure where a variable changes from 0 to 1 over time.*
*curve: The animation curve of this procedure.* See **SKCurve.**
*time: Total time of this procedure.*
*action: Action is called every frame during this procedure. It is provided with a float variable from 0 to 1.*
*onFinish: The funtion called after the procedure is finished.*
*id: ID for this procedure. Use it to stop the procedure later.*

csharp

```
//Bounce a ball from left to right in 2 seconds.
SKUtils.StartProcedure(SKCurve.BounceIn, 2.0f, (t)=>
{
```

```
    ball.position = Vector3.Lerp(left, right, t); // t moves from 0 to 1
});
```

csharp

```
//Fade in a image, then after 5 seconds, fade out.
SKUtils.StartProcedure(SKCurve.LinearIn, 1.0f, (t)=> //fade in
{
    image.SetAlpha(t); // t moves from 0 to 1
},()=> //onFinish
{
    SKUtils.InvokeAction(5.0f, ()=> //wait for 5 seconds
    {
        SKUtils.StartProcedure(SKCurve.LinearOut, 1.0f, (t)=> //fade out
        {
            image.SetAlpha(t);// t moves from 1 to 0 (because the curve is
        });
    });
});
```

** There are also some other overloads of this function. Please refer to the inline comments for further information.*

**void StopProcedure(string id)**

*Stops an ongoing StartProcedure call specified by its id.*

## 1.5 Object Pooling

SKCell has a built-in object pool system. Be sure to have the **SKPoolManager** component in your scene.

**GameObject SpawnObject(GameObject go)**

*- Spawn an object using the built-in object pool.*

*- returns: The spawned instance.*

**void ReleaseObject(GameObject go)**

*- Release ("destroy") an object in the built-in object pool.*

**void ReleaseObject(GameObject obj, float time)**

*- Release ("destroy") an object after some seconds.*

csharp

```
//EXAMPLE
GameObject fx_prefab;
GameObject fx_instance = SKUtils.SpawnObject(fx_prefab);
//...
SKUtils.ReleaseObject(fx_instance);
```

## 1.6 Destroying & Disabling

**void Destroy(GameObject go)**

*Correctly destroys a game object in both play and edit modes. Skips if the object does not exist.*

**void SetActiveEfficiently(GameObject go, bool enabled)**

*Skips if the game object is already enables/disabled.*

**void DeactivateByTeleport(GameObject go)**

*Instead of disabling a game object, send it to a far location. Doing this is more efficient in most cases.*

**void ReactivateTeleportedObject(GameObject go)**

*Restore the teleported game object to its original position.*

## 1.7 Input

These functions require the **SKInput** component.

**void AddMouseDownAction(int mouseID, Action a)**

*Register an action on event of mouse down.*

```csharp
//EXAMPLE
void Start()
{
    SKUtils.AddMouseDownAction(0,Jump);
}
void Jump(){}

//This is same as:
void Update()
{
    if(Input.GetMouseButtonDown(0))
```

```
        {
            Jump();
        }
    }
    void Jump(){}
```

**void RemoveMouseDownAction(int mouseID, Action a)**

**void AddMouseUpAction(int mouseID, Action a)**

**void RemoveMouseUpAction(int mouseID, Action a)**

**void AddKeyDownAction(KeyCode kc, Action a)**

**void RemoveKeyDownAction(KeyCode kc, Action a)**

**void AddKeyUpAction(KeyCode kc, Action a)**

**void RemoveKeyUpAction(KeyCode kc, Action a)**

## 1.8 Graphics

**void GLDrawLine(Vector3 v1, Vector3 v2, int drawMode = GL.LINES)**
*Draw a line using low-level graphics library.*

**void GLDrawTriangle(Vector3 v1, Vector3 v2, Vector3 v3, int drawMode = GL.TRIANGLES)**
*Draw a triangle using low-level graphics library.*

**void GLDrawQuads(Vector3 v1, Vector3 v2, Vector3 v3, Vector3 v4, int drawMode = GL.QUADS)**
*Draw a quad using low-level graphics library.*

**void MeshNormalAverage(Mesh mesh)**
*Average the normals of a mesh. (in-place)*
**This can be used in smooth outlining effects.*

**void CombineMeshes(GameObject ori, GameObject tar)**

*Combine two meshes into one.*

**[Legacy] There are several custom mesh related functions.*

## 1.9 Base & Data

**T[] Serialize2DArray(T[,] arr)**
*Serialize a 2D array into a 1D array.*

**T[,] Deserialize2DArray(T[] arr, int len1, int len2)**
*Deserialize a 1D array into a 2D array.*
*len1: row count.*
*len2: column count.*

csharp

```
//2D array
{{1,2,3},
{4,5,6}}
//Serialized 1D array
{1,2,3,4,5,6}
```

** This is for when 2D arrays are not compatible with certain serialization methods.*

**T[] ModifyArray(T[] arr, int length)**
**T[,] Modify2DArray(T[,] arr, int width, int height)**
*Change the length of an array while preserving its contents.*

**bool InsertOrUpdateKeyValueInDictionary<TKey, TValue>(Dictionary<TKey, TValue> dict, TKey key, TValue value)**

*Insert a key value pair into a dictionary. If the key already exists, update the value.*

**bool RemoveKeyInDictionary<TKey, TValue>(Dictionary<TKey, TValue> dict, TKey key)**

*Remove key in dictionary only if the key exists.*

**TValue GetValueInDictionary<TKey, TValue>(Dictionary<TKey, TValue> dict, TKey key)**

*Get value in dictionary only if the key exists.*

**void InsertToList(List list, T item, bool allowMultiple)**

*Insert an item into a list.*

*allowMultiple: If not, duplicate items will not be inserted.*

**void RemoveFromList(List list, T item)**

*Remove an item from a list only if the item exists.*

**void SwapValue(ref T value1, ref T value2)**

*Swap two values.*

**void FillArray(T[] arr, T item)**

*Fill the array with the item.*

**int CountInArray(T[] arr, T item)**

*Get the count of item in array.*

**T[] RemoveDuplicatesInArray(T[] arr)**

*Remove duplicates in array.*

**float MaxInArray(float[] arr)**

*Get the max float in array.*

**float MinInArray(float[] arr)**

*Get the min float in array.*

**string CompressString(string str)**

*Compress a string. (UTF-8 + GZipStream)*

**string DecompressString(string str)**

*Decompress a compressed string.*

## 1.10 Math

**Vector3 Angle2Vector(float angle)**
*Angle (degrees) to 2D vector. (e.g. 90 deg -> Vector2(1,0))*

**int Vector2Angle(Vector3 dir)**
*2D vector to angle (degrees), rounded to the nearest integer. (e.g. Vector2(-1,0) -> 180 deg)*

**Vector3 ApplyRotationToVector(Vector3 vec, Vector3 vecRotation)**
*Rotate a vector.*
*vecRotation: rotation given by direction vector.*

**float SampleCurve(SKCurve curve, float x)**
*Sample an animation curve specified by an* **SKCurve.**
*x: time of the sample. (0-1)*

**float GetAngle(Vector3 selfDirection, Vector3 compareDirection)**
*Get the angle between two vectors.(-180 deg, 180 deg)*

**float GetAngleXZPlane(Vector3 selfDirection, Vector3 compareDirection)**
*Get the angle between two vectors on the XZ plane.*

**bool LinesIntersect2D(Vector3 ptStart0, Vector3 ptEnd0, Vector3 ptStart1, Vector3 ptEnd1,bool firstIsSegment, bool secondIsSegment)**
*Test if two 2D lines intersect.*
*firstIsSegment: does the first line has limited length?*
*secondIsSegment: does the second line has limited length?*

**bool LinesIntersect2D(Vector3 ptStart0, Vector3 ptEnd0, Vector3 ptStart1, Vector3 ptEnd1, bool firstIsSegment, bool secondIsSegment,ref Vector3 pIntersectionPt)**
*Test if two 2D lines intersect and get the intersection point.*

**Vector3 LineIntersectPlane(Vector3 point, Vector3 direct, Vector3 planeNormal, Vector3 planePoint)**
*Get the intersection point between a line and a plane.*

**(bool intersect, Vector3 p1, Vector3 p2) LineIntersectSphere(Vector3 start, Vector3 end, Vector3 center, float radius)**
*Get the intersection point between a line and a sphere.*

**bool PointInRectangle(Vector3 rectCenter, Vector3 rectSize, float rotation, Vector3 point)**
*Is the point inside the given rectangle?*

**void CalcRectangleRotate(Vector3 rectCenter, Vector3 rectSize, float rotation,out Vector3 p1r, out Vector3 p2r, out Vector3 p3r, out Vector3 p4r)**
*Returns a rectangle rotated by rotation degrees.*

**bool LayerContains(LayerMask mask, int layer)**
*Does the layerMask contain the given layer?*

**int CalcStrCRC32(string p_str)**
*return Animator.StringToHash(p_str);*

**int HashKey(Transform transform)**
*Get the hash key of a transform. (only on windows)*

**bool IsPrime(int n)**
*Is the given int n a prime?*

**float Keep2Decimal(float num)**
*Round float to 2 decimal points.*

**float Keep2Decimal(float num)**
*Round float to 2 decimal points.*

**float ColorLuminance(Color c)**
*Get the luminance value of a color.*

## 1.11 Reflection

These reflection functions are quite self-explanatory.

**FieldInfo[] GetConstants(System.Type type)**

**void CallMethod(string method, Type type, object param)**

**Type GetSpecificType(Type type)**

**List GetDerivedTypes(Type baseType)**

**bool IsSubclassOf(Type type, Type baseType)**

**bool IsSubclassOf(Type type, Type baseType)**

## 1.12 I/O

**void SaveObjectToJson(object obj, string fileName)**
*Serialize and save an object to Json file. Default path is root/json/.*

**T LoadObjectFromJson(string fileName)**
*Deserialize a json file. Default path is root/json/.*

**bool JsonFileExists(string fileName)**
*Test if the json file exists. Default path is root/json/.*

**[Legacy] There are several SK versions of these functions.*

## 2. Extensions

Lots of useful extension methods are available to you.

### 2.1 Transform

**void ResetTransform(this Transform tf, bool isLocal)**
*Reset the transform components to default.*
*isLocal: if true, reset local position, rotation, and scale.*

**void ResetPosition(this Transform tf, bool isLocal)**
*Reset the transform position to default.*

**void ResetLocalScale(this Transform tf)**
*Reset the transform local scale to default.*

**void ResetLocalRotation(this Transform tf)**
*Reset the transform local rotation to default.*

**void ResetGlobalRotation(this Transform tf)**

*Reset the transform global rotation to default.*

**void SetPositionX(this Transform tf)**

*Set the X component of the position.*

csharp

```
//instead of this
transform.position = new Vector3(20,transform.position.y,transform.position
//use this!
transform.SetPositionX(20);
```

**void SetPositionY(this Transform tf)**

**void SetPositionZ(this Transform tf)**

**void SetRotationX(this Transform tf)**

**void SetRotationY(this Transform tf)**

**void SetRotationZ(this Transform tf)**

**void SetScaleX(this Transform tf)**

**void SetScaleY(this Transform tf)**

**void SetScaleZ(this Transform tf)**

**void Get2DPosition(this Transform tf)**

*return new Vector3(tf.position.x, tf.position.y, 0);*

**void CopyFrom(this Transform selfTf, Transform otherTf)**
*Set the transform value from another transform.*

**List GetAllChildren(this Transform tf)**
*Return an list of all the children of the given transform. (recursively)*

**void ClearChildren(this Transform tf)**
*Destroy all children.*

**void ClearChildrenImmediate(this Transform tf)**
*DestroyImmediate all children. (edit mode)*

**Transform CreateChild(this Transform tf)**
*Add a child with default transform values.*

**void SwapSiblingOrder(this Transform selfTf, Transform otherTf)**
*Swap sibling order with another sibling.*

**void SwapSiblingOrder(this Transform selfTf, Transform otherTf)**
*Swap sibling order with another sibling.*

## 2.2 RectTransform

**bool Contains(this RectTransform a, RectTransform b)**
*Check if a ReectTransform contains another.*

**bool Overlaps(this RectTransform a, RectTransform b)**

*Check if a ReectTransform overlaps another.*

**Rect WorldRect(this RectTransform rectTransform)**

*Get a Rect in world space from a RectTransform.*

## 2.3 Animator

**void Appear(this Animator anim)**

*anim.SetBool("Appear", true);*

**void Disappear(this Animator anim)**

*anim.SetBool("Appear", false);*

**void Pop(this Animator anim)**

*anim.SetTrigger("Pop");*

These are only by some conventions.

csharp

```
//instead of this
anim.SetBool("Appear", true);
//use this!
anim.Appear();
```

## 2.4 Texture

**void SetColor(this Texture2D t, Color color)**

*Set the color of the entire Texture2D.*

**void SetQuad(this Texture2D t, Vector2Int lb, Vector2Int rt, Color color)**

*Set the color of a rectangular region of pixels in a Texture2D.*

**void AddQuad(this Texture2D t, Vector2Int lb, Vector2Int rt, Color color)**

*Add to the color of a rectangular region of pixels in a Texture2D.*

**void MultiplyQuad(this Texture2D t, Vector2Int lb, Vector2Int rt, Color color)**

*Multiply to the color of a rectangular region of pixels in a Texture2D.*

```csharp
//EXAMPLE
texture.SetQuad(new Vector2Int(0,0), new VectorInt(512,512),Color.Cyan);
```

*Be sure to apply the changes after calling these functions!*

## 2.5 Data

**Vector2Int ToVector2Int(this Vector2 v)**

*Cast a Vector2 to a Vector2Int.*

**List PopulateList(this List list, T content, int count)**

*Populate the list with a certain number of an item.*

**T[] PopulateArray(this T[] array, T content, int count)**

*Populate the array with a certain number of an item.*

**float SimpleDistance(this Vector3 v, Vector3 v1)**

*Manhattan distance between vectors.*

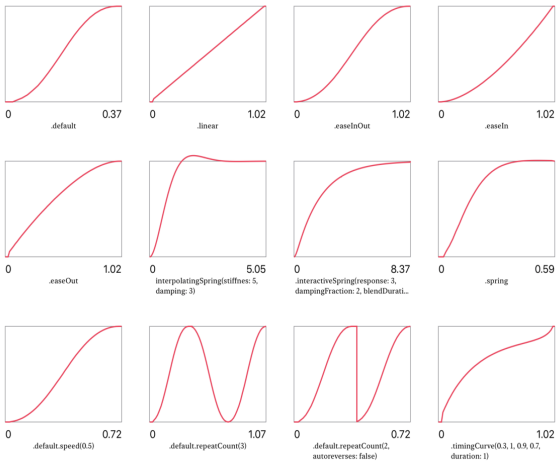
**float SimpleDistanceSigned(this Vector3 v, Vector3 v1)**

*Manhattan distance between vectors where the order of vectors matters.*

# 3. Structures

## 3.1 SKCurve

**SKCurve provides an easy way to access various animation curves.**




Available built-in curves include:
- Linear
- Quadratic
- Cubic
- Quartic
- Quintic
- Sine
- Exponential
- Elastic
- Circular
- Bounce
- Back

All of the curve values are **from 0 to 1**.
To get a curve, use SKCurve.xxx:

```csharp
SKCurve.CubicDoubleIn
SKCurve.BounceOut
SKCurve.LinearIn
```

### Suffixes

- The **Double** suffix indicates animation on both ends of the curve.
- The **In** and **Out** suffixes specifies the direction of the curve. **In** is from 0 to 1, while **Out** is from 1 to 0.

### Sampling

To sample a curve, use the **SKCurveSampler.SampleCurve** function:

```csharp
//sample the SineIn curve at x=0.7
float y = SKCurveSampler.SampleCurve(SKCurve.SineIn, 0.7f);
```

## 3.2 Singletons

### Non-Monobehaviour Singletons

To make your class a singleton (non-Monobehaviour), inherit from the **Singleton** class:

```csharp
using SKCell;

public class GameManager : Singleton<GameManager>
{

}
```

Then you can access this singleton using

```csharp
GameManager.instance
```

### Monobehaviour Singletons

To make your class a Monobehaviour singleton, inherit from the **MonoSingleton** class:

```csharp
using SKCell;

public class Boss : MonoSingleton<Boss>
{

}
```

Then you can access this singleton using

```csharp
Boss.instance
```

*Note: If you would like to implement an Awake function in your singleton, be sure to override it.*

## 3.3 Tree Node

TreeNode is a tree structure with these fields for each node:

- name (string)
- parent (TreeNode)
- children (List)
- value (generic)

### Methods

**void AddChild(ITreeNode child)**
*Add a child.*

**void AddChild(int index, ITreeNode child)**
*Add a child at the given index.*

**void ClearChild()**
*Clear all children.*

**ITreeNode[] GetAllChild()**

*Get all immediate children as an array.*
**ITreeNode[] GetAllChildRecursive()**
*Get all children in the tree rooted at current node as an array.*

**ITreeNode GetChild(int index)**
*Get a child at the given index.*

**int GetChildIndex(ITreeNode child)**
*Get the index of a given child.*

**bool HasChild(string name)**
*Does the current node have a child with the given name?*

**void RemoveChild(int index)**
*Remove the child at the given index.*

**void RemoveChild(string name)**
*Remove the child with the given name.*

**void SetParent(ITreeNode parent)**
*Set the parent of the current node.*

**void DetachFromParent()**
*Detach from the current parent.*

### Tree Structure

The TreeStructure class gives a naive implementation of a tree using Tree Nodes.
It has the following fields:

csharp

```
    private string name;
    private ITreeNode<T> headNode;
    private ITreeNode<T>[] nodeList;
```

*Please refer to the code for more details.*

## 3.4 Priority Queue

You can find a priority queue implementation in the PriorityQueue class.

Specify the order of the elements by setting **isDescending** when constructing:

```csharp
public PriorityQueue(bool isdesc) : this()
{
    IsDescending = isdesc;
}
```

### Methods

#### void Enqueue(T x)

#### T Dequeue()

#### T Peek()

#### void Clear()

## 3.5 Command Pattern

### Command

The **Command** class is an abstract class representing a single command.
Override the **Execute** and **Revert** methods in your own implementation.

csharp

```
public class MyCommand:Command
{
    public override void Execute(params float[] args){
        //...
    }
    public override void Revert(params float[] args){
        //...
    }
}
```

### CommandManager

The **CommandManager** class keeps a command stack and provide important functions for the command pattern to work.

After creating a Command, push it onto the stack by calling

csharp

```
CommandManager.StackPush(command, args);
```

To repeat the last command, call

```csharp
CommandManager.Do();
```

To undo the last command, call

```csharp
CommandManager.Undo();
```

You can get the last command and its parameters with

```csharp
CommandManager.LastCommand
```

## 3.6 Bezier Curve

The **SKBezier** class provides a simple representation of 3-Bezier curves.

Each curve includes 2 endpoints and 2 control points:

p0: first endpoint

p1: first control point

p2: second endpoint

p3: second control point

### Functions

**Vector3 Sample(float t)**
*Sample the curve at the given time (0-1). Returns the sampled position.*

**Vector3[] Path(int segments)**
*Get the path representation of the curve. Divide the curve into certain number of segments and return the turn points.*

**float Length(int pointCount = 15)**
*Get the length of the path. Higher point count gives higher accuracy.*

**Vector3 Tangent(float t)**
*Get the tangent vector of the path (facing direction) at the given time.*

# 4. Events

The event system allows you to register to and execute events from anywhere without coupling.

## 4.1 Event Reference

You can give each event an integer ID using the **EventRef** class. There are 9 preset regions of event labels.

csharp

```
//EXAMPLE
public static readonly int CM_ON_SCENE_LOADED = 1000;
public static readonly int CM_ON_SCENE_EXIT = 1001;
//...
public static readonly int UI_CONV_ON_NEXT_SENTENCE = 5100;
public static readonly int UI_CONV_ON_SELECT_OPTION = 5101;
```

## 4.2 Event Dispatcher

The **EventDispatcher** class manages registration and dispatching of events.

### Adding a listener

To add a listener to an event, use the **EventDispatcher.AddListener** method: **bool AddListener(EventHandler handler, int id, SJEvent t_event)**

csharp

```
//EXAMPLE
void Start()
{
    EventDispatcher.AddListener(EventDispatcher.Common, EventRef.CM_ON_SCEN
}
```

*handler: There are 9 preset event handlers in EventDispatcher. Use the corresponding handler by typing EventDispatcher.xxx.*

*id: This is the event reference you created in EventRef.*

*t_event: The function you want to register to this event.*

### Dispatching an event

To dispatch an event, use the **EventDispatcher.Dispatch** method: **void Dispatch(EventHandler handler, int id)**

csharp

```
//EXAMPLE
void OnSceneLoaded()
{
    EventDispatcher.Dispatch(EventDispatcher.Common, EventRef.CM_ON_SCENE_L
}
```

Everything registered onto the event will be executed.

## 4.3 Variable Monitor

The **SKVariableMonitor** object can monitor a variable and tell you when it changes.

csharp

```
//EXAMPLE
float speed;

SKVariableMonitor<float> speedMonitor = new SKVariableMonitor<float>(()=>{
    return speed; //Here is to provide a access point of your monitered var
});
speedMonitor.onValueChanged += OnSpeedChange;

private void OnSpeedChange(float spd){
```

```
    //Do something when speed changes
    }
```

# 5. UI

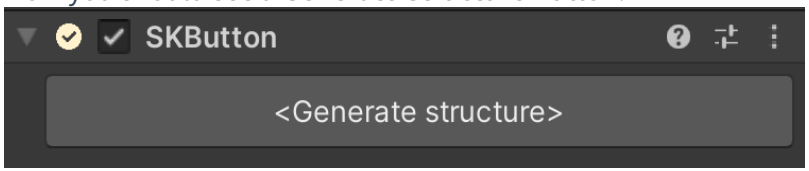
SKCell has a powerful UI system built upon Unity's UGUI.

## 5.1 SKButton

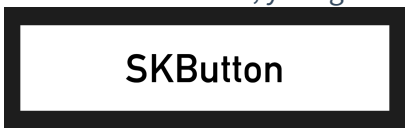
SKButton is a customizable, flexible button UI component with built-in animation & events without the need to code.

### Creating an SKButton

Inside a canvas, add a new game object. Then, attach the SKButton component onto it.
Now you should see a **Generate Structure** Button:




Click on that button, your game object will be replaced by an SKButton preset.




Now try play it! This button should have mouse over and click animations.

### Customizing SKButton

**Interactable**
If turned off, the button will not respond to any events.

**Button Image & Text**
References. Do not need to change these in most circumstances.

**Transition Mode**
There are 6 transition modes. The default is Color Image And Text.

- Color Image Only
- Color Text Only
- Color Text Only
- Color Image And Text
- Animation
- None

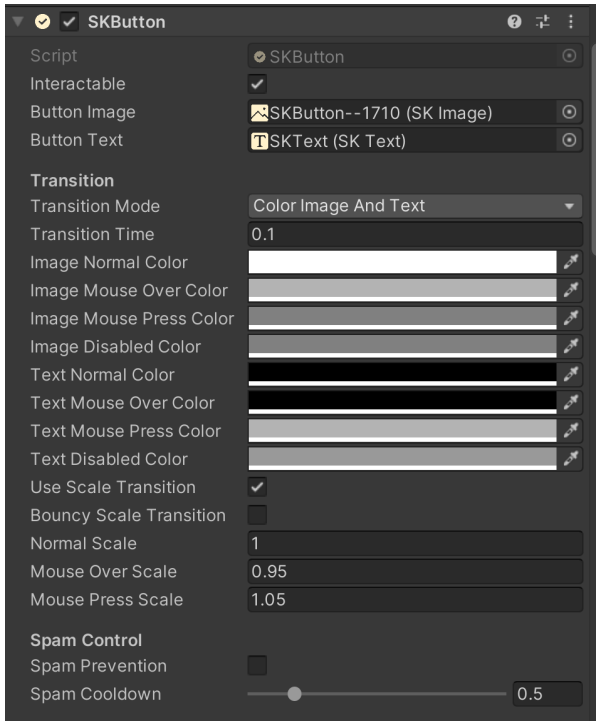
**Transition Time**
Time of a single transition (sec).

**Normal / Over / Press Colors**
Set the colors here and the button will do the transition for you.

**Use Scale Transition**
Do you need scale animation for transitions?

**Normal / Over / Press Scales**
Set the scales here and the button will do the transition for you.

**Spam Control**
Prevent the user from clicking the button multiple times in a short time.




## Button Events

These are the events available. Register to them either from the editor or call the **SKButton.AddListener** method.
- OnPress
- OnHoldStays
- OnHoldStaysForSeconds
- OnHoldStaysForSeconds2
- OnHoldUp
- OnHoldUp2
- OnPointerEnter
- OnPointerExit
- OnPointerUp
- OnPointerDown
- OnStart

csharp

```
//Register a press & hold event (3 seconds)
myButton.holdForSecondsTime = 3.0f;
myButton.AddListener(SKButtonEventType.OnHoldStaysForSeconds, myFunc);

//Use current hold time to add visual effects
float sec = myButton.GetCurrentHoldTime();
FillSlider(sec/3.0f);
```

## Methods

**void AddListener(SKButtonEventType type, UnityAction action)**
*Add a listener to an SKButton event.*

**void RemoveListener(SKButtonEventType type, UnityAction action)**
*Remove a listener from an SKButton event.*

**void RemoveAllListeners(SKButtonEventType type)**
*Remove all listeners from an SKButton event.*

**void SetText(string text)**
*Set the button text directly.*

**void UpdateText(int localID)**
*Set the button text using the **Localization system**.*

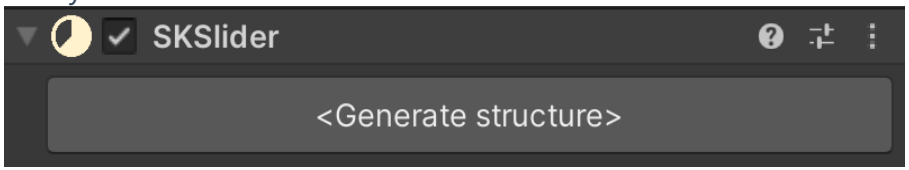
**float GetCurrentHoldTime()**
*Get the current hold time of this SKButton. Return 0 if not being held.*

## 5.2 SKSlider

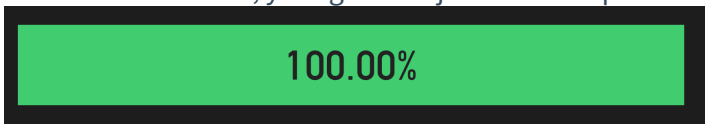
SKSlider is a customizable, flexible progress bar UI component with built-in animation & events without the need to code.

### Creating an SKSlider

Inside a canvas, add a new game object. Then, attach the SKSlider component onto it.
Now you should see a **Generate Structure** Button:



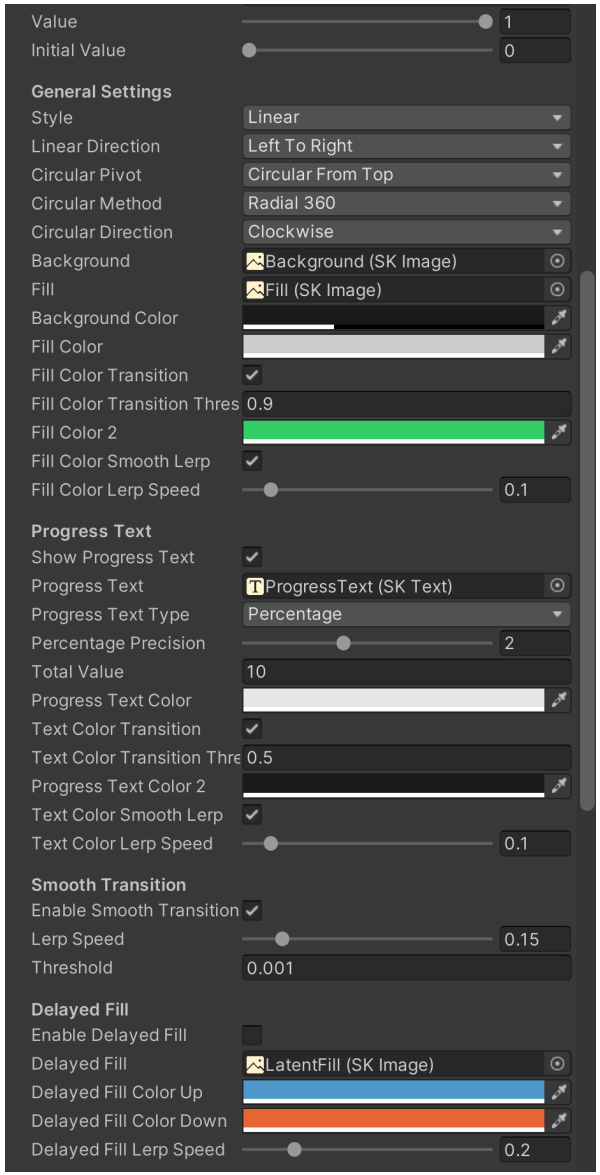

Click on that button, your game object will be replaced by an SKSlider preset.




Now try adjusting its value in the editor!

### Customizing SKSlider

Be sure to click **Generate Structure** again to apply changes after you modify the style / structure of the slider!




### Value
Value of the slider. (0-1)

### Initial Value
Value of the slider on start. (0-1)

### Style
There are 2 slider styles:
- Linear
- Circular

### Linear Direction
Direction for linear style.

### Circular Pivot / Method / Direction
Starting point / fill method / fill direction for circular style.

### Slider Background & Fill
References. Do not need to change these in most circumstances.

### Fill Color
Color of fill.

### Fill Color Transition
If turned on, fill color will change if slider value is over some threhold.

### Fill Color Transition Threshold
Fill color will change if slider value is over this threhold.

### Fill Color 2
Second fill color. The color to change into if

slider value is over the threhold.

**Fill Color Smooth Lerp**
Smoothly or abruptly transition into fill color 2?

**Fill Color Lerp Speed**
Speed of smooth transition into fill color 2. Default is 0.1.

**Progress Text Type** There are 2 progress text types:
- Percentage (e.g. 95.01%)
- Part by Whole (e.g. 5/12)

**Percentage Precision**
Number digits after decimal point. (Percentage Mode)

**Total Value**
Total value. (Part by Whole Mode)

**Text Color / Transition / Color 2 / Smooth Lerp / Lerp Speed**
Same as those for fill color.

**Smooth Transition**
If turned on, slider value will change smoothly.

**Lerp Speed / Threshold**
Speed of smooth transition / threshold for the lerp (leave it as default in most circumstances).

**Delayed Fill**
Common effect involving a second fill area behind the fill that moves slower.

**Delayed Fill Color Up**
Color of delayed fill when the value increases.
(e.g. health restore)

**Delayed Fill Color Down**
Color of delayed fill when the value decreases.
(e.g. injury)

**Delayed Fill Lerp Speed**
Speed for delayed fill. Default is 0.2.

### Slider Events

These are the events available. Register to them from the editor. - OnValueChanges
- OnFull
- OnEmpty
- OnHalf
- OnStart

### Methods

**void SetValue(float value)**
*Set the value of this SKSlider. Visual effects will be applied.*

**void SetValueRaw(float value)**
*Set the value of this SKSlider. Visual effects will NOT be applied.*

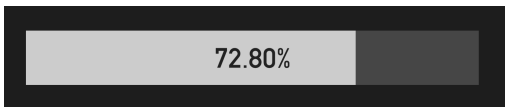
**void SetRandomValue()**
*Set a random value for this SKSlider.*

csharp

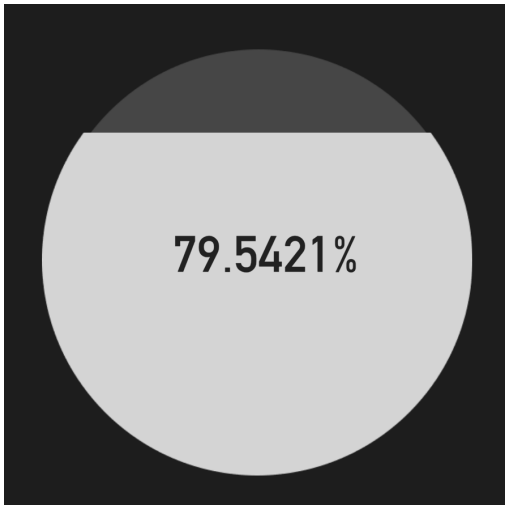
```
//Set value of 0.2 to the slider.
mySlider.SetValue(0.2f);
```

**Difference styles an SKSlider can have:**

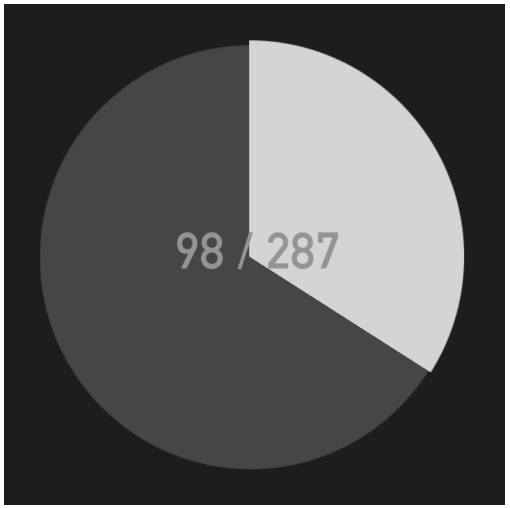
Linear + Left to Right + Percentage Text + 2 Precision




Circular + Bottom to Top + Percentage Text + 4 Precision




Circular + Circular from Top + Radial 360 + Part by Whole Text

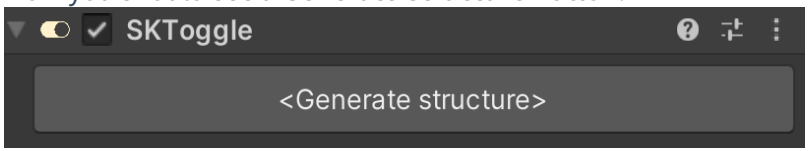



## 5.3 SKToggle

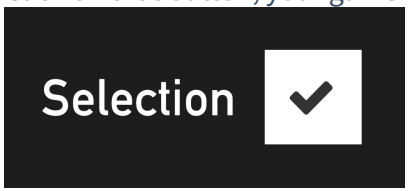
SKToggle is a customizable, flexible toggle UI component with built-in animation & events without the need to code.

### Creating an SKToggle

Inside a canvas, add a new game object. Then, attach the SKToggle component onto it.
Now you should see a **Generate Structure** Button:




Click on that button, your game object will be replaced by an SKToggle preset.




Now try play it! This toggle should have animations and interactivity.

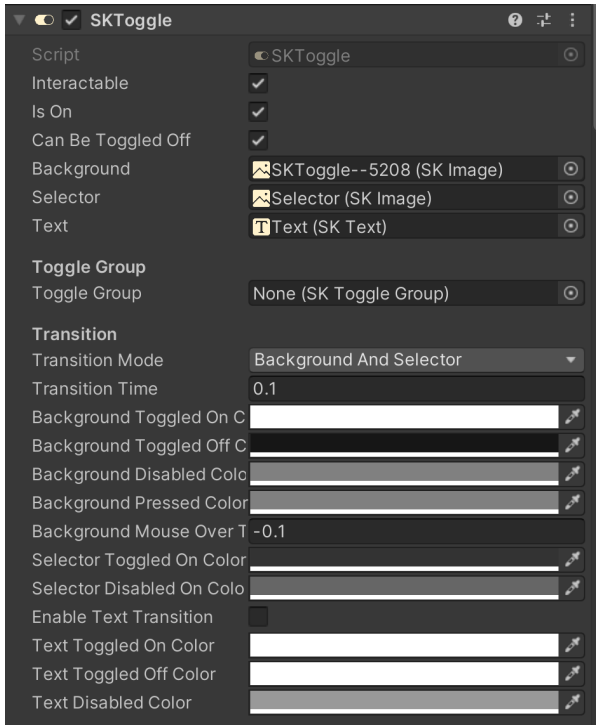
### Customizing SKToggle

**Interactable**
If turned off, the toggle will not respond to any events.

**IsOn**
Current status of the toggle.

**Can be toggled off**
If on, the toggle will keep being on and not interactable.

**Background & Selector & Text**
References. Do not need to change these in most circumstances.




**Toggle Group**
The Toggle Group this toggle belongs to. See SKToggleGroup.

**Transition Mode**
There are 4 transition modes. The default is Background And Selector.
- Animation
- Background Only
- Selector Only
- Background And Selector

**Background Colors**
Color of background in various states.

**Selector Colors**
Color of selector in various states.

**Text Colors**
Color of text in various states.

**Enable Text Transition**
If off, text color will not change during transition.

## Toggle Events

These are the events available. Register to them either from the editor or call the **SKToggle.AddListener** method.
- OnToggled
- OnToggledOn
- OnToggledOff
- OnPress
- OnPointerEnter
- OnPointerExit
- OnPointerUp
- OnPointerDown
- OnStart

csharp

```
//Disable a game object when the toggle is toggled off.
myToggle.AddListener(SKToggleEventType.OnToggledOff, ()=>
{
    go.SetActive(false);
});
```

## Methods

**void Toggle()**
*Toggle this toggle.*

**void ToggleOn()**
*Toggle this toggle on.*

**void ToggleOff()**
*Toggle this toggle off.*

**void AddListener(SKToggleEventType type, UnityAction action)**
*Add a listener to an SKToggle event.*

**void RemoveListener(SKToggleEventType type, UnityAction action)**
*Remove a listener from an SKToggle event.*

**void RemoveFromGroup()**
*Remove this toggle from the toggle group.*

## 5.4 SKToggleGroup

SKToggleGroup clusters multiple SKToggles together with some collective behavior.

### Setting Up SKToggleGroup

1. Prepare several toggles
2. Create a new SKToggle Group
3. Drag the toggle group to each toggle
4. Adjust toggle group settings for your needs

### Customizing SKToggleGroup

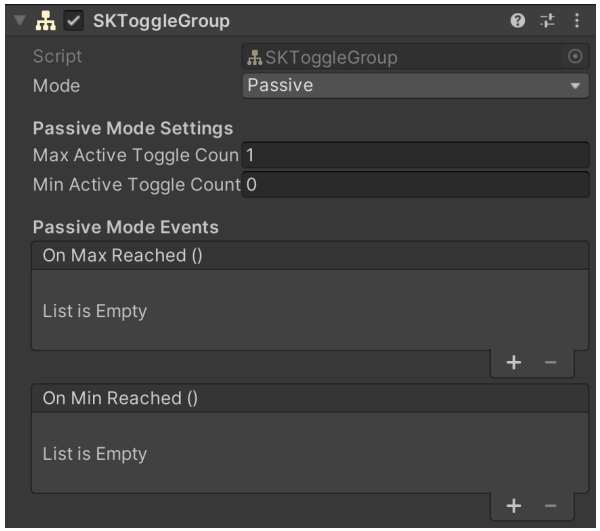
**Mode**
- Passive: No special actions.

- Active One Only: There can only be 1 active toggle in this group.

**Passive Mode Settings**

Max Active Toggle Count
Min Active Toggle Count
*Toggle group will satisfy these constraints automatically.*

**Passive Mode Events**



There are 2 events.
- On Max Reached: called when max active toggle count is reached.
- On Min Reached: called when min active toggle count is reached.

## 5.5 SKText

SKText is an extended component from TextMeshPro's Text. It is integrated with the text animator and the localization system.

See **SKLocalization** and **SKTextAnimator** for more details.

### Methods

**void UpdateTextDirectly(string newText)**
*Update text without **localization**.*

**void UpdateText(T arg0)**
**void UpdateText(T arg0, T arg1)**
**void UpdateText(T arg0, T arg1, T arg2)**
*Update text with given arguments for **localization**.*

**void UpdateLocalID( int localID )**
*Update the **local ID** for this text.*

### Usage Example

To use SKText with localization:
1. Set up some text entries in the **Localization Control Center**
2. Note down the **local ID** of the text you want to display (e.g. 3012)
3. Call:

```csharp
skText.UpdateLocalID(3012);
```

Then the text will update to be what local ID 3012 corresponds to.

To use SKText with text animator:

1. Find the SKTextAnimator component attached to this SKText.
2. Go to **SKTextAnimator** for more details.

Inline text effects are active as default.
For example, type **"<wave>This is an example text.</>"** into the text box will apply wave effect to the text.

## 5.6 SKImage

SKImage is an extended component from UGUI's Image. It is integrated with the localization system.
See **SKLocalization** for more details.

### Methods

**void UpdateImageDirectly(Sprite sprite)**
*Update image without **localization**.*

**void UpdateLocalID( int localID )**
*Update the **local ID** for this image.*

### Usage

Similar to SKText. The only difference is you need to set up an image localization in the **Localization Control Center**.

## 5.7 SKUIAnimation

SKUIAnimation gives multiple common transition animations to your UI.
These animations are two-state, meaning that there are only 2 clips: one for active, one for inactive.

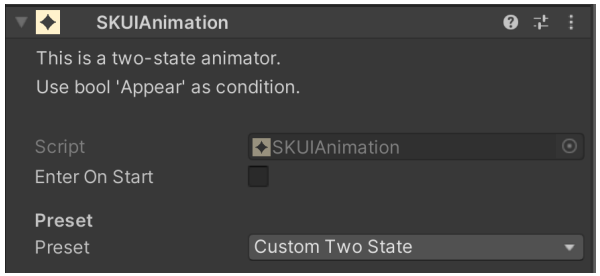
Here are the available presets:

- InstantAppear
- AlphaFade
- FadeLeft
- FadeRight
- FadeUp
- FadeDown
- ScaleUp
- ScaleDown
- AlphaFadeSlow
- InstantInFadeOut
- Shine

To preview these effects, enter play mode and you can find 2 buttons in the inspector: **Preview State: Enter** and **Preview State: Exit**.

**Enter On Start**
Is the state **Enter** or **Exit** on start?

**Preset**
The desired preset.




## Using SKUIAnimation

It is extremely simple!

1. Add SKUIAnimation component to a UI object.
2. Select preset.
3. Play!

Use the following method to control the animation:

csharp

```
uiAnim.SetState(true);
//or
uiAnim.SetState(false);
```

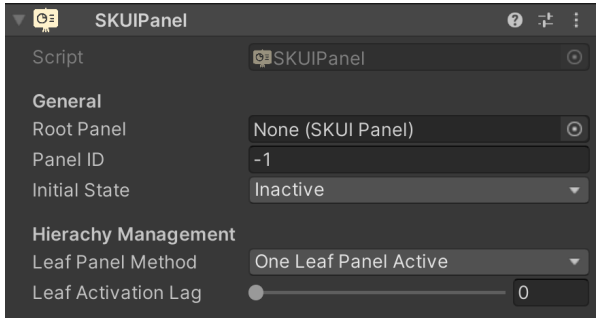
Then the transition animation will play!

## Methods

**void SetState(bool appear)**
*Set the state of this animation. Play the corresponding state.*

## 5.8 SKUIPanel

SKUIPanel represents a panel with animation, children, and parent structures.




**Root Panel**
Parent panel of this panel. This panel will be added as leaf on start.

**PanelID**
Integer ID of this panel.

**Initial State**
State of this panel on start. Will affect the SKUIAnimation component correspondingly.

**Leaf Panel Method**
Method to manage leaf panels.

**Currently there is only one method: one leaf panel active. This option will keep only one leaf panel in its active state at the same time. You can create menu tabs with this.*

### Fields

**int lastActivatedLeaf**
*ID of the last activated leaf.*

**List leafPanels**
*List of leaf panels.*

### Methods

**void ActivateLeaf(SKUIPanel leaf)**
**void ActivateLeaf(int leafPanelID)**
*Set state of a leaf panel as active.*

**DeactivateLeaf(int leafPanelID)**
*Set state of a leaf panel as inactive.*

**void SetState(SKUIPanelState state)**
*Set state of current panel.*

## 5.9 SKUIPanelManager

SKUIPanelManager manages all the SKUIPanels in the game.

### Panel Hierarchy

There are 7 preset panel hierarchies. They differ by their sorting order.
- UILowermost -- 1
- UILow -- 2
- UIMain -- 3
- UIHigh -- 4
- UIHigher -- 5
- UITopmost -- 6
- UIConstant -- 7

To create a UI root and view those hierarchies, select **Tools/SKCell/UI/BulidPanelHeirarchy** in the menu bar.

### Methods

**void ActivatePanel(int panelID)**
**void ActivatePanel(SKUIPanel panel)**
*Set state of a panel as active.*

**void DeactivatePanel(int panelID)**
**void DeactivatePanel(SKUIPanel panel)**
*Set state of a panel as inactive.*

**int GetActivePanelCount()**
*Get count of currently active panels.*

**SKUIPanel GetPanelByID(int id)**
*Get panel by ID.*

**bool? IsActive(int panelID)**
*Is the given panel active? Returns null if panel does not exist.*

**void InstantiatePanel(int panelID, SKUIPanelState state, int predecessor = -1, Transform parent = null)**
*Instantiate a new SKUIPanel with the given info.*
*predecessor: the new panel will be instantiated one level lower in heirarchy than the predecessor.*
*parent: transform parent of the new panel.*

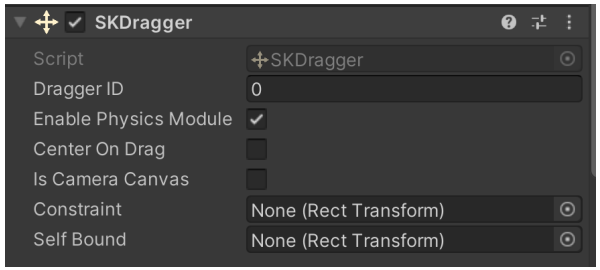
**public void InstantiatePanel(int panelID, SKUIPanelState state, SKUIPanelHierarchy hierarchy)**
*Instantiate a new SKUIPanel in the given panel hierarchy.*

**void DisposePanel(int panelID)**
*Destroy a given panel.*

**void DisposePanel(int panelID)**
*Destroy a given panel.*

## 5.10 SKDragger

SKDragger adds dragging functionalities to a UI component.




**Dragger ID**
Integer ID of this SKDragger.

**Enable Physics Module**
If on, 2D physics will be involved. This makes drag constraints, stickers, spawners, etc. possible. Default is on.

**Center On Drag**
Center the object at mouse position on drag. Default is off.

**Is Camera Canvas**
Select if this object exists in a camera canvas.

**Constraint**
Reference to a constraint rect transform. Dragged objects will not be allowed to go beyond the constraint area.

**Self Bound**
Reference to a self bound rect transform. This is self area of this object. Use this rect transform as default.

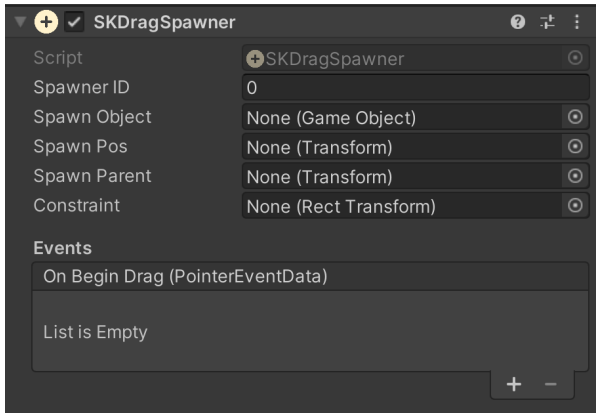
**Events**

**OnBeginDrag**
**OnDrag**
**OnEndDrag**
**OnEnterConstraint**
**OnLeaveConstraint**

These two events are tied with SKDragSpawner and SKDragSticker.
**OnSpawn**
**OnDispose**

## 5.11 SKDragSpawner

SKDragSpawner allows you to "drag out" a draggable UI object. (e.g. dragging objects in an inventory)




**Spawner ID**
Integer ID of this SKDragSpawner.

**Spawn Object**
The object you want to spawn. *Be sure to have the SKDragger component on the spawn object!*

**Spawn Pos**
Position to spawn. Use this transform as default.

**Spawn Pos**
Parent of spawned object. Use this transform as default.

**Constraint**
Constraint of the spawn object.

**Events**

## 5.12 SKDragSticker
**OnBeginDrag**

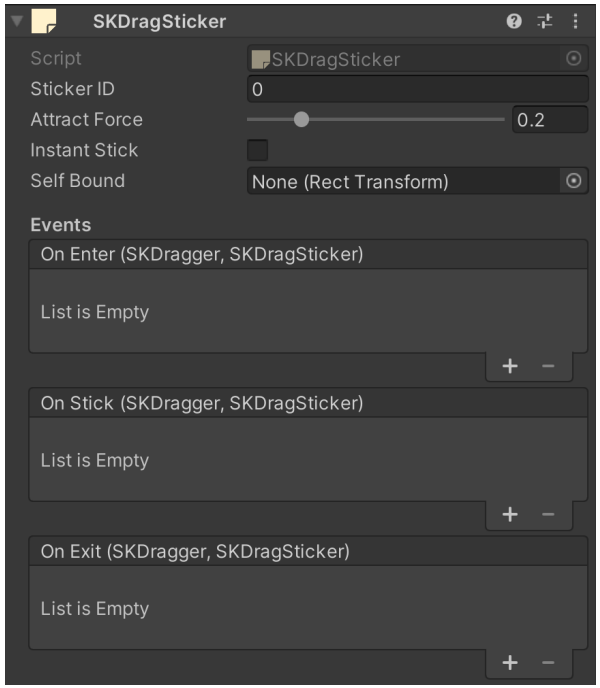
SKDragSticker attracts draggable objects, can think of this as a "destination".

**Sticker ID**
Integer ID of this SKDragSticker.

**Attract Force**
Force with which this sticker attracts draggers.
Default is 0.2.

**Instant Stick**
If on, draggers will be instantly placed onto the sticker without smooth transition.

**Self Bound**
Effective area of this sticker.




**Events**

**OnEnter**
**OnStick**
**OnExit**

**Usage**

Try use SKDragger, SKDragSpawner, SKDragSticker to make a simple draggable inventory!

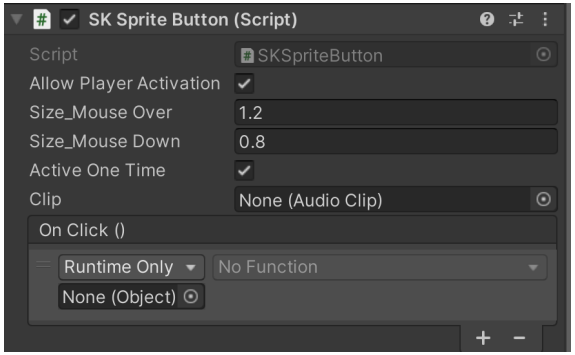
## 5.13 SKSpriteButton

SKSpriteButton adds button-like functionalities to a sprite. No UI components are needed.

## Usage

To use SKSpriteButton,

1. Find an object with SpriteRenderer as the button.
2. Attach SKSpriteButton onto it.
3. Add a collider on the object as the interactable area.
4. Assign OnClick events if needed.
5. Play!




### Allow Player Activation

*[Deprecated]*

### Size Mouse Over

Size of button when mouse is over.

### Size Mouse Down

Size of button when mouse is pressed.

### Active One Time

If on, the button can be only pressed once.

### Clip

Audio clip to play when clicked.

## 5.14 SKTranslationalSlider

A slider using image translation instead of image fill.

Common usage include a translatable fill image inside a mask.

### Fields

**Slider Content**
The object to slide.

**Start Pos & End Pos**
The slider content will move from start pos to end pos. (start pos when value = 0, end pos when value =1)

**Progress Text**
Text to display slider value as percentage. (2-precision)

**Value**
Value of slider. (0-1)

## 5.15 SKUIModelViewer

SKUIModelViewer offers a template for displaying and interacting with 3D models in UI panels.

### Usage

To use SKUIModelViewer,

1. Create a new game object and add the SKUIModelViewer component.
2. Click the **Generate Structure** button.

To customize the model,

3. Expand the generated game object.
4. Adjust the camera and the model to display you own scene.

#### Interaction

SKUIModelViewer comes with a gallery-like display where you can rotate the displayed model by dragging the viewer.
To customize this behaviour, adjust the parameters under the **Gallery Behaviour** section.

### 5.16 Example Scene

**You can see more examples in the SKUIScene.**

## 6. Effects

### 6.1 SKImageProcessing

SKImageProcessing offers common image effects such as saturation, contrast, brightness, color shift, etc.

#### Using SKImageProcessing

1. Add the component onto your image.
2. Adjust the parameters!

## Supported Effects

**Original Image**

**Customizable Blend Mode**
Src Blend = OneMinusDstColor
Dst Blend = OneMinusSrcColor

**Hue Shift**
Hue shift = 0.8f

**Brightness**
Brightness = 2.0f

**Saturation**
Saturation = 0.0f

**Contrast**
Contrast = 1.5f

**Alpha Fading**
Fade from bottom

**Alpha Fading**
Fade from right

**Fading from all directions is available.*

You can stack these effects on top of each other! For example, you can let the image fade from left while having a saturation value of 2.0f.

Example to access this at runtime:

csharp

```
const float BRIGHTNESS = 0.7f;
SKImageProcessing ip = go.GetComponent<SKImageProcessing>();
//Gradually increase go's brightness in 2 seconds
SKUtils.StartProcedure(SKCurve.LinearIn, 2.0f, (t)=>
{
    ip.brightness = 1 + t * BRIGHTNESS;
});
```

Reference to **SKUtils.StartProcedure**: **Tweening**.

## 6.2 SKSpriteProcessing

This is a sprite version of **SKImageProcessing**. Functions are indentical.

## 6.3 SKBackBlur

Back blur effect is commonly used in games, such as the blurring of the scene when you open a menu.

SKBackBlur makes it extremely easy to implement such effects.

### Using SKBackBlur

1. Add the component onto any image. The image will act as the area of blur.
2. Place the image onto the area you want to blur! Be sure the image is **in front of** what you want to blur.

### Preview

**Original Image**

**Blur = 3.0f**

**Blur = 12.0f**

You can also add additive & multiplicative tint to the blurred area.

## 6.4 SKSpriteBlur

Unlike SKBackBlur that blurs the background, SKSpriteBlur blurs the image it is attached onto.

**Preview**

**Original Image**

**Blur = 0.15f**

**Blur = 0.8f**

Notice that this effect does not look really good for large blurs. I will try to fix this in later versions!

## 6.5 Light2D

*[Under Development] I plan to release this as a separate system. Currently the component is usable but not efficient.*

## 6.6 SKUIParticleSystem

SKUIParticleSystem allows particle systems to be displayed as UI.

### Using SKUIParticleSystem

1. Add the component onto a UI object.
2. A particle system component will be added automatically. Work with it like a normal particle system!

### Remarks

Check the settings in the inspector carefully.
Be sure to adjust the "Max Particle Count" property! If too low, many of your particles will not be visible.

## 6.7 SKAnimationRandomizer

SKAnimationRandomizer randomizes the start time of an animation clip on play. A useful scenario would be where you have hundreds of grass objects with the same sway animation, and you want them to start asynchoronously so that it looks natural.

### Using SKAnimationRandomizer

1. Add the component onto an object with an animator.
2. Play!

## 6.8 SKUIShadow

SKUIShadow lets you add drop shadow effects to your UI elements.

### Using SKUIShadow

1. Add the SKUIShadow component to your UI object.
2. Adjust the color, spread, iterations, etc. to fit your needs.

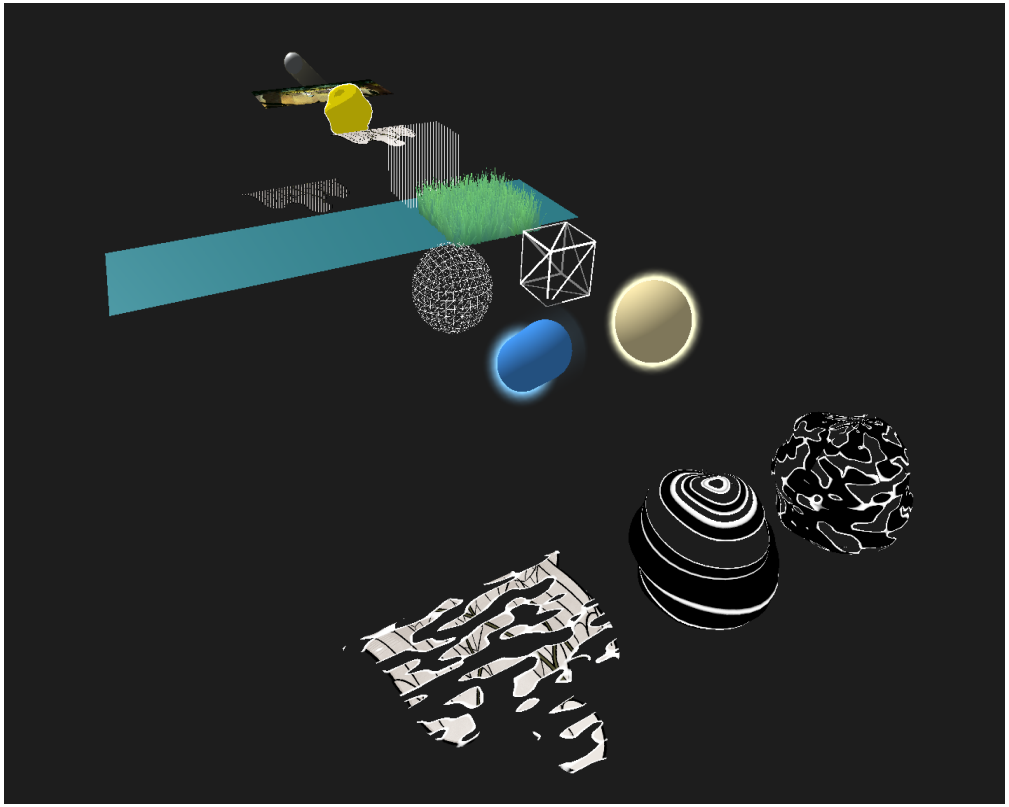
## 6.9 SKLineAnim

SKLineAnim is an interesting example of how you can make a animated curve.

### Using SKLineAnim

1. Go to SKCell/Effects and place the LineAnim prefab into your scene.
2. Observe!

## 6.10 Many More Shader Effects!



There are many more effects which include: SKDissolveEffect, SKOuterGlowEffect, SKCartoonGrass, SKToonMaterial, SKGlitchEffect, SKWireframeEffect, SKDitherAlpha, SKEdgeOutlineEffect, SKLightCastEffect, etc.
You can find then in the **SKShaderScene**.

## 7. Text Animator

SKCell offers a dynamic inline text animator which allows you to implement text effects just by marking the texts.
It also comes with built-in typewriter effects.
For example,

**<dangle>Okay!**
**</> Let's see...**


will be rendered

as -->
**<wave>Impressive!**
**</>**
will be rendered
as -->

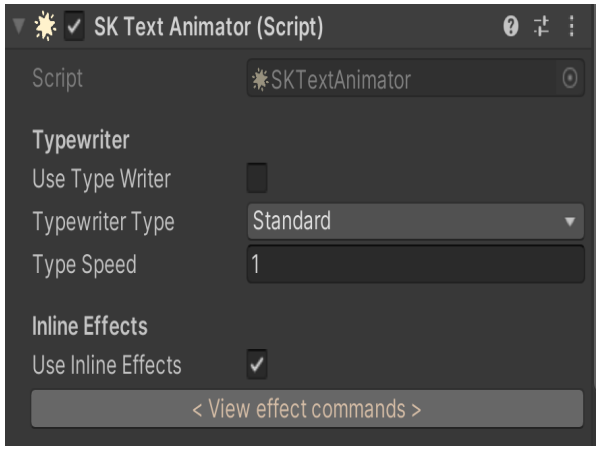

**These effects are dynamic. Be sure to preview them in your editor!**

## 7.1 Getting Started

SKTextAnimator will be automatically added when you use **SKText**.

**After adding the SKText component, you should see an SKTextAnimator component as well.**

There are two major effects: **Typewriter** and **Inline Effects**.




## 7.2 Typewriter Effects

To activate typewriter effects, simply check **Use Typewriter** in the inspector.

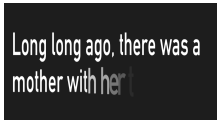
Hit play now, the text should appear in a beautiful typewriter effect.
Then you can adjust the **style** and **speed** to your needs.

If you want to replay the effect later, call

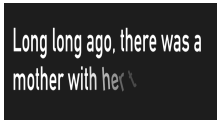
```csharp
textAnimator.PlayTypeWriter();
```

### Example Effects

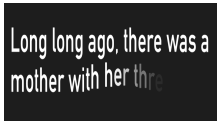
**Standard Typewriter**
Alpha fade + scaling




**Rotate Typewriter**
Alpha fade + rotation




**Wave Typewriter**
Alpha fade + wave




### All Presets

- Standard

- Alpha Fade

- Rotate

- Wave

- Translate

- Shake

### Fast Forward

To fast forward a typewriter effect, call

```csharp
textAnimator.TypewriterFastForward();
```

## 7.3 Inline Text Effects

You can easily add text effects to part of your text using the following syntax:

<EffectName & Parameters>Your Text </>
Simple examples are shown **here**.

### Effect Presets

You can view these by selecting **View Effect Commands** in the inspector.

```
To use command: <effectName(arg0,arg1...)> your text </>
1. Shake: <shake>       <shake(time)>
2. Banner: <banner(time,r,g,b)>
3. Fade: <fade>         <fade(speed)>
4. Twinkle: <twinkle>         <twinkle(speed)>
5. Dangle: <dangle>         <dangle(speed)>
6. Exclaim: <excl>         <excl(speed,r,g,b)>
7. Timed Exclaim: <exclt>         <exclt(speed,time,r,g,b)>
8. Wave: <wave>         <wave(speed)>
9. Scale Up: <scaleup>         <scaleup(speed)>
10. Scale Down: <scaledn>         <scaledn(speed)>
11. Rotate: <rot(speed,angle)>
12. Color: <col(speed,time,r,g,b)>
```

### Simple Inline Effects

These are preset effects with fixed parameters. You can access these using only the effect name. That means, they do not accept any parameters.

**- Shake: <shake>**
*This effect shakes the characters individually at a high speed. Looping. Can (possibly) express: freezing, chill, horror, surprise, etc.*

**- Fade: <fade>**
*This effect fades out the characters individually. Non-looping. Can (possibly) express: secret, embarassment, etc.*

**- Twinkle: <twinkle>**
*This effect fades in and out the characters individually. Looping. Can (possibly) express: highlighting, treasure, etc.*

**- Dangle: <dangle>**
*This effect dangles the characters. Non-looping. Can (possibly) express: dejected, discouraged, disappointment, etc.*

**- Exclaim: <excl>**
*This effect makes the characters "scream" (scale-up & shaking & color change). Looping. Can (possibly) express: excitement, surprise, etc.*

**- Timed Exclaim: <exclt>**
*This is exclaim but non-looping. Will stop in a short time.*

**- Wave: <wave>**
*This effect makes the characters flow up and down. Looping. Can (possibly) express: relaxation, humor, etc.*

**- Scale Up: <scaleup>**
*This effect makes the characters bigger. Non-looping. Can (possibly) express: emphasis, highlighting, etc.*

**- Scale Down: <scaleup>**
*This effect makes the characters smaller. Non-looping.*

Example:

```
Wow, you got the <excl>MEDAL</>?
<dangle>This can't be true...</> I'm <shake>furious</>!
<wave>Hi, how's your day?</>
```

### Parameterized Inline Effects

You can customize most of the effects by passing in parameters.
The syntax is:

```
<name(p1,p2,p3,...)>Your Text</>
```

For example,

```
//shake for 5 seconds
<shake(5)>Behold, the crown of Maloth!</>

//exclaim at 2x speed with the color of red
<excl(2,1,0,0)>Stop there, invader.</>
```

Details about these can be found in the inspector.

## 7.4 Remarks

There are currently some limitations of this system.

- Multiple effects on the same block of text is **NOT** supported. For example,

```
//This is invalid
<wave><excl>Some Text</></>
```

- Be sure to follow the syntax precisely. For example,

```
//These are invalid because no space is allowed inside of <>
<shake (5.3) >Some Text</>
<shake( 5.3)>Some Text</>
```

- Do not use the **"<"** character except for inline effects. This can result in a parse error. **Will fix in later versions!*

# 8. Dialogue System

SKCell has a node editor system for creating and displaying dialogues. You do not need to write any code to implement simple dialogues.

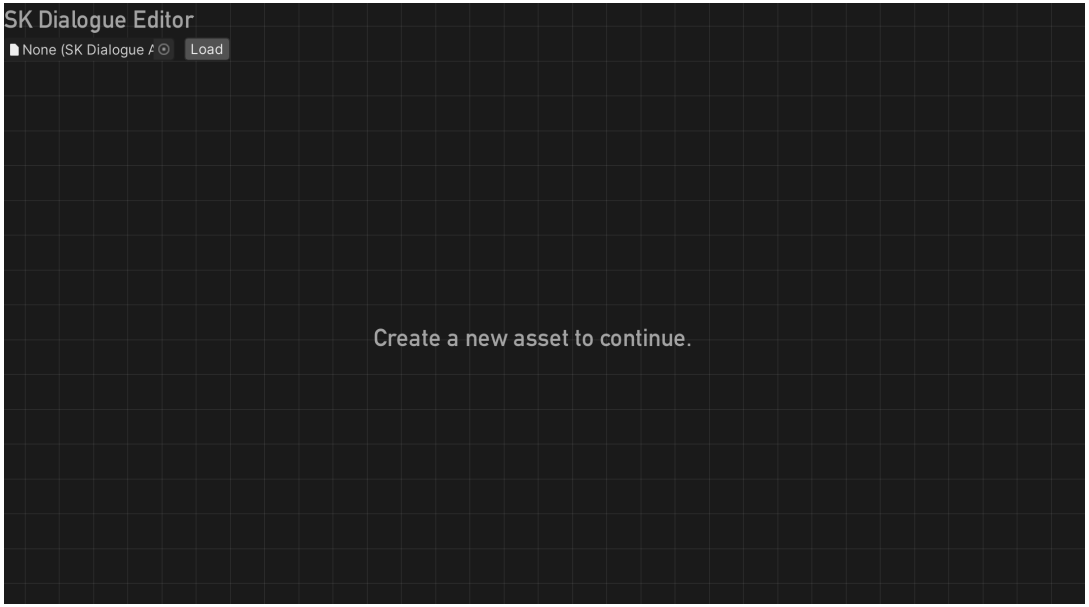
## 8.1 Getting Started

### How the dialogue system works

- Each dialogue is contained in an **SKDialogueAsset**.
- You can edit the asset using the **SKDialogueEditor**.
- Then the dialogue can be displayed using the **SKDialoguePlayer**.

### Opening the Dialogue Editor

Select **Tools/SKCell/Dialogue Editor** from the menu bar.
You will be directed to the following window.
.




### Creating a Dialogue Asset

In your project window, right click, select **Create/SKCell/Dialogue Asset**.

### Loading a Dialogue Asset

To load a dialogue asset, drag the asset to the top-left corner of the dialogue editor (or select it directly from the object field), then press **Load**.

### Editing a Dialogue Asset

See **Dialogue Editor**.

### Playing a Dialogue Asset

See **Dialogue Player**.

## 8.2 SK Dialogue Editor

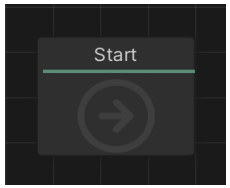
The dialogue editor allows to to make dialogues by connecting nodes.

### Dialogue Flow

The dialogue will start with the **Start Node**. Then it will follow the connection you make to various nodes while executing those nodes along the way.
At last, the dialogue will end either when reaching a deadend or the **End Node**.

### Node Types

**1. Start Node**
The dialogue starts here. There can only be one start node in a dialogue.



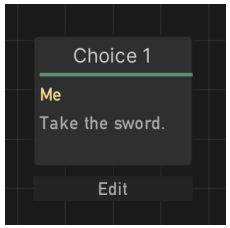

**2. End Node**
The dialogue ends here. When the end node is reached, the dialogue will stop.
There can be multiple end nodes.


**3. Sentence Node**
This represents a sentence. When a sentence node is reached, the correspond sentence will be displayed.


**Editing a Sentence Node**
Open the edit panel by clicking **Edit** at the bottom of the node.
You can assign speaker name / local ID

(localized), content text / local ID (localized), audio clip, and avatar to the sentence.

**4. Choice Node**



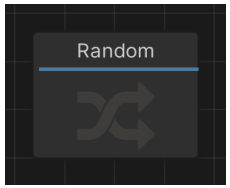

This represents a choice after a sentence. When a choice node is reached, several selectable choices will be displayed. The dialogue will not continue until a choice is selected by the player.

**Editing a Choice Node**

Open the edit panel by clicking **Edit** at the bottom of the node.

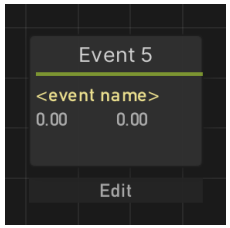
You can assign speaker name / local ID (localized) and content text / local ID (localized) to the choice.

**5. Random Node**




When the random node is reached, the dialogue will randomly proceed to one of its connected nodes.

**6. Event Node**




When the event node is reached, an event will be executed with a name (identifier), and 2 float arguments.

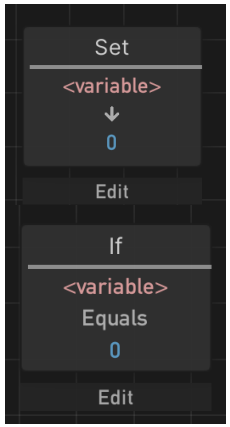
Use

**void SKDialoguePlayer.AddListenerToEvent(string eventName, System.Action callback)**

to register to these events.

For example, you can add an item to player's inventory after a certain progression into the dialogue using event nodes.

**7. Set Node**




When the set node is reached, the system will set the value of a variable defined by you.

**8. If Node**

When the if node is reached, a comparison involving a variable will be performed. If it passes, the dialogue moves on, if not, the dialogue ends here.

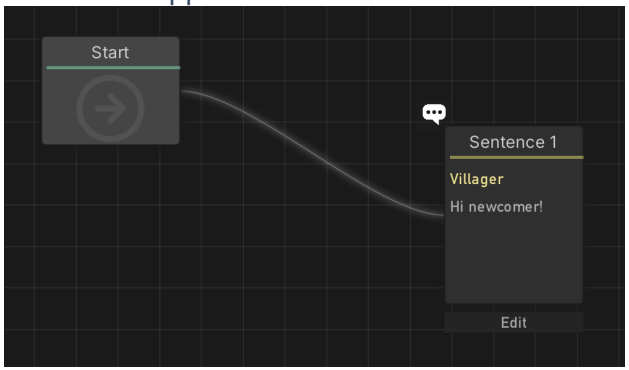
## Creating Nodes

Right click on an empty area and select **Create New Node/...** to create a new node.

You can also right click on an existing node and and select **Create New Node/...** to create a new node connected from the existing node.

## Connecting Nodes

To connect two nodes, right click on the first node, select **Link to...**, and left click on the second node. A curve will appear between these nodes.

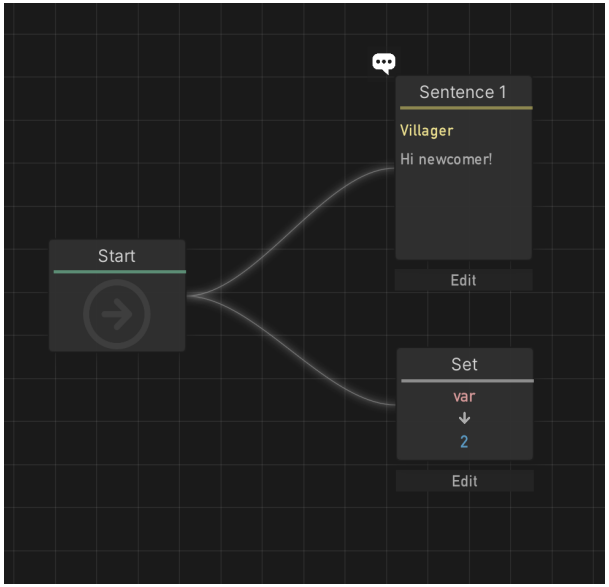



**Links are directed! Source and destination nodes are not exchangeable.*

To delete a connection, right click on the source node and select **Unlink.../** to unlink a specific connection.

## Multiple Links

You can connect multiple nodes to one node.

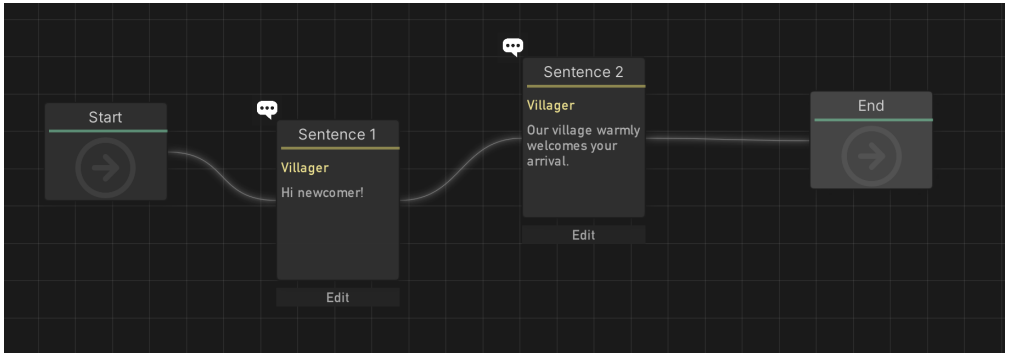



In this case, all connected nodes will be **executed at once**.
**Because of this, do not connect multiple sentences to a single node (except for the random node). It will not make sense.*

## 8.3 Dialogue Flow Examples

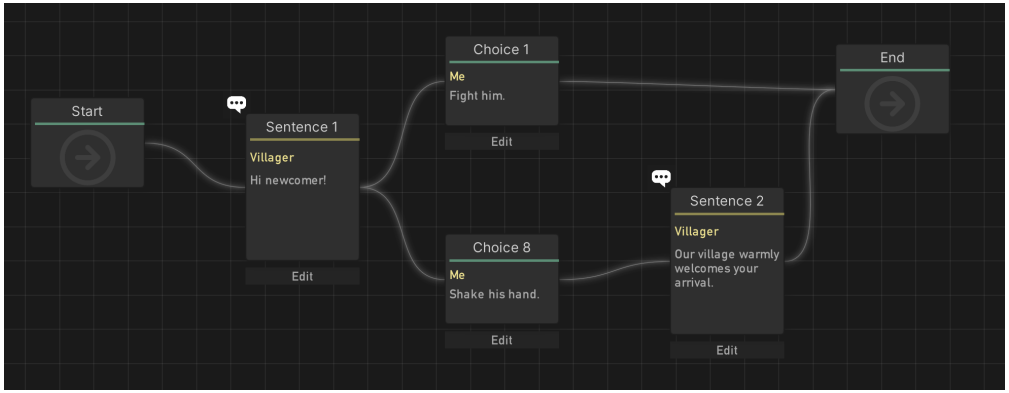
Here are some useful examples to further illustrate how the editor works.

### 1. Simple Linear Dialogue



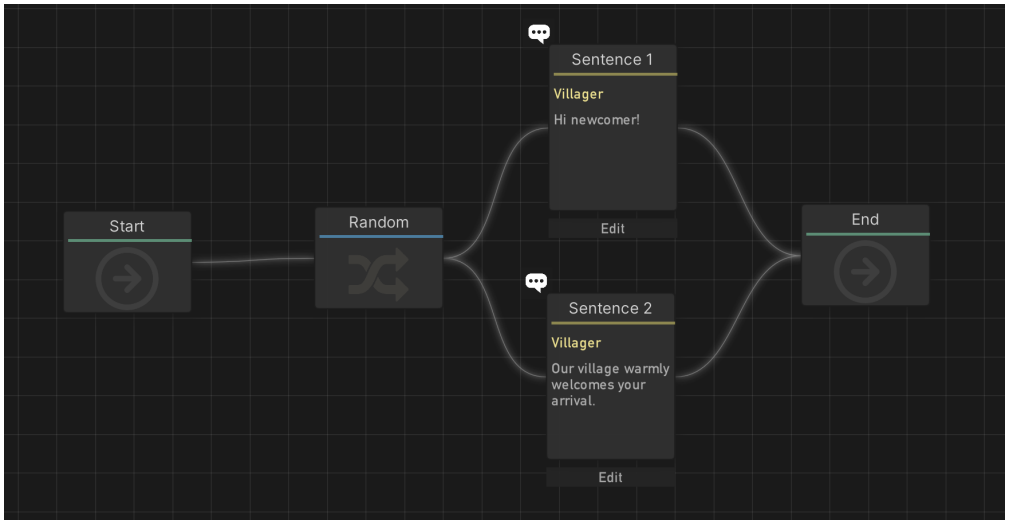
To display some linear sentence sequence, connect them in order between start and end nodes.

### 2. Branching Choices



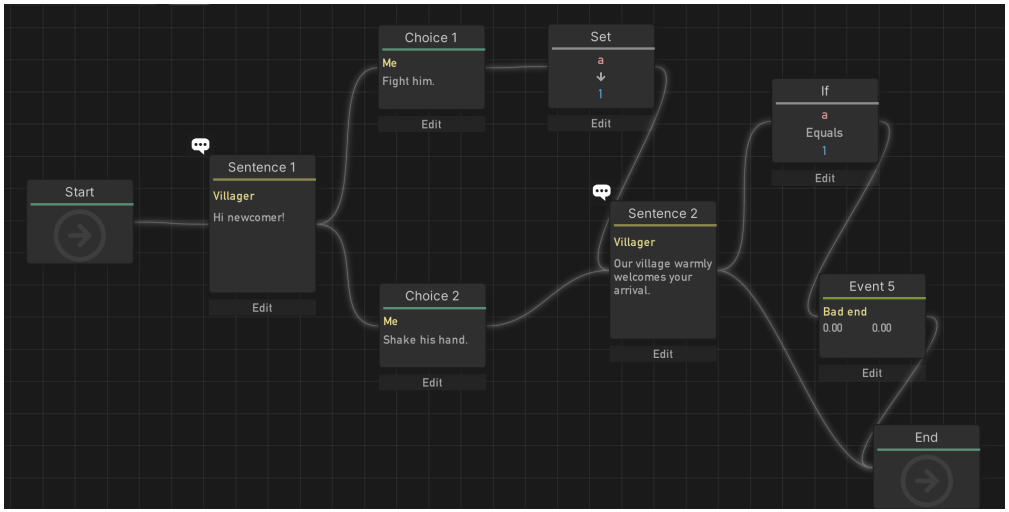

The dialogue will go differently depending on the choices the player makes.

### 3. Random



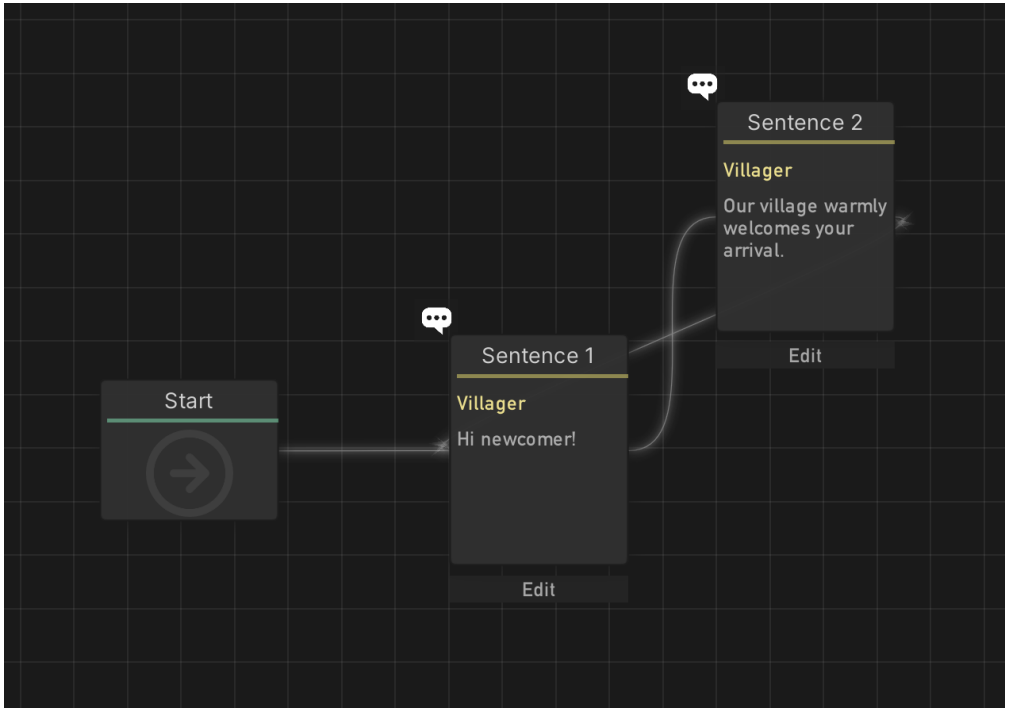

The dialogue will randomly display one of the sentences.

## 4. Branching Variables




The dialogue will go differently according to the variable.

## 5. Loops




You can make connections to previous parts of the dialogue. In this example, the dialogue will never end.

## 8.4 SK Dialogue Player

SKDialoguePlayer is the component that actually plays the dialogue asset.

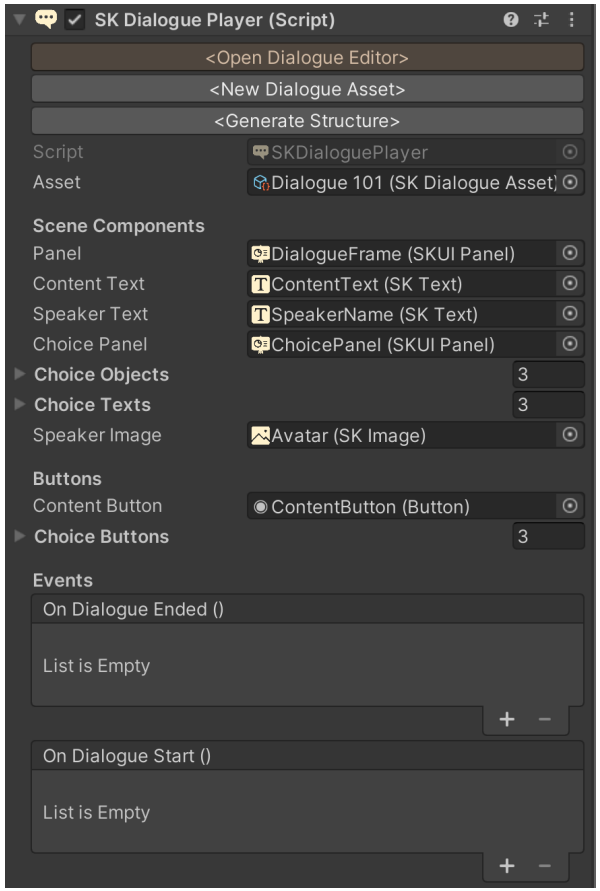
You can find a prefab of the structure required by SKDialoguePlayer in **Tools/SKCell/Resources/SKCell/Prefabs/SKDialogue**.

### To Use SKDialoguePlayer

1. Create a new game object.
2. Add the SKDialoguePlayer component.
3. Click **Generate Structure**.

4. Assign the dialogue asset.
5. Play! (either by selecting **Play on start** or calling **Play()**)

## Structure




### Generate Structure
This generate a preset structure for the Dialogue Player to work. It will assign the requires references automatically. You can customize it afterwards.

### Asset
The dialogue asset to play.

### Scene Components
### Panel
The overall dialogue panel (**SKUIPanel**). This mainly controls the appearing / disappearing of the panel.

### Content & Speaker Text
Text components for displaying the content and the speaker. (**SKText**).
Thanks to the **SKTextAnimator** functionalities that comes with SKText, you have typewriter effects automatically implemented!

### Choice Panel
The panel to display selectable choice buttons (**SKUIPanel**).

### Choice Objects & Texts
References to the actual choice objects. The default count is 3; if you want more or fewer choices, feel free to modify this.

**Speaker Image**
SKImage component that displays speaker's avatar.

**Content Button**
The button component that allows players to continue with the dialogue. (click for the next sentence) Normally this has the size of the dialogue frame.

## Events

**Choice Buttons**
The button components for each of the choice objects.

**OnDialogueStart**
**OnDialogueEnded**

## Methods

**void Play()**
*Play the assigned dialogue asset.*

**void Play(SKDialogueAsset asset)**
*Play a dialogue asset.*

**void SentenceNextStep()**
*Continue the dialogue flow by one step. (e.g. clicking on the sentence text -> fast forward / go to next sentence)*

**void AddListenerToEvent(string eventName, System.Action callback)**
*Register to one of the events in the dialogue.*
*eventName: the name you specify in the dialogue editor.*
*callback: the function you want to call. The two float parameters correspond to what you specify in the dialogue editor.*

csharp

```
//EXAMPLE
SKDialoguePlayer player;
```

```
player.AddListenerToEvent("On Trade Complete", (id,count)=>
{
    AddItem(id, count);
});
```

### 8.5 Example Scene

**You can see more examples in the SKDialogueScene.**

## 9. Localization

SKCell has a easy-to-use localization system for texts and images.

*Audio localization will be available in later versions.*

#### How It Works

- All text and image entries are identified by a unique **Local ID.**
- Each entry includes content for multiple languages.
- Text and image components will read from the entries according to the currently active language.

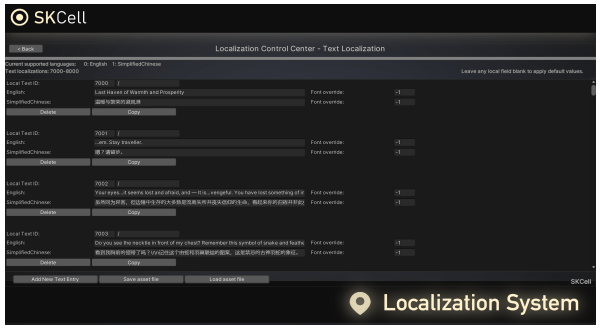
### 9.1 Local Entries & Local ID

#### Local Entries

> **Text Entry (example)**
>
> Local Text ID: local ID for this text entry.

Comments: you can type in comments next to the local ID. Comments are only there for reference.

Language fields: you can type in the content for the supported languages here.




### Local ID

Local IDs are unique identifiers each corresponding to a local entry. This is the only way to access local entries.

### Accessing Local Entries

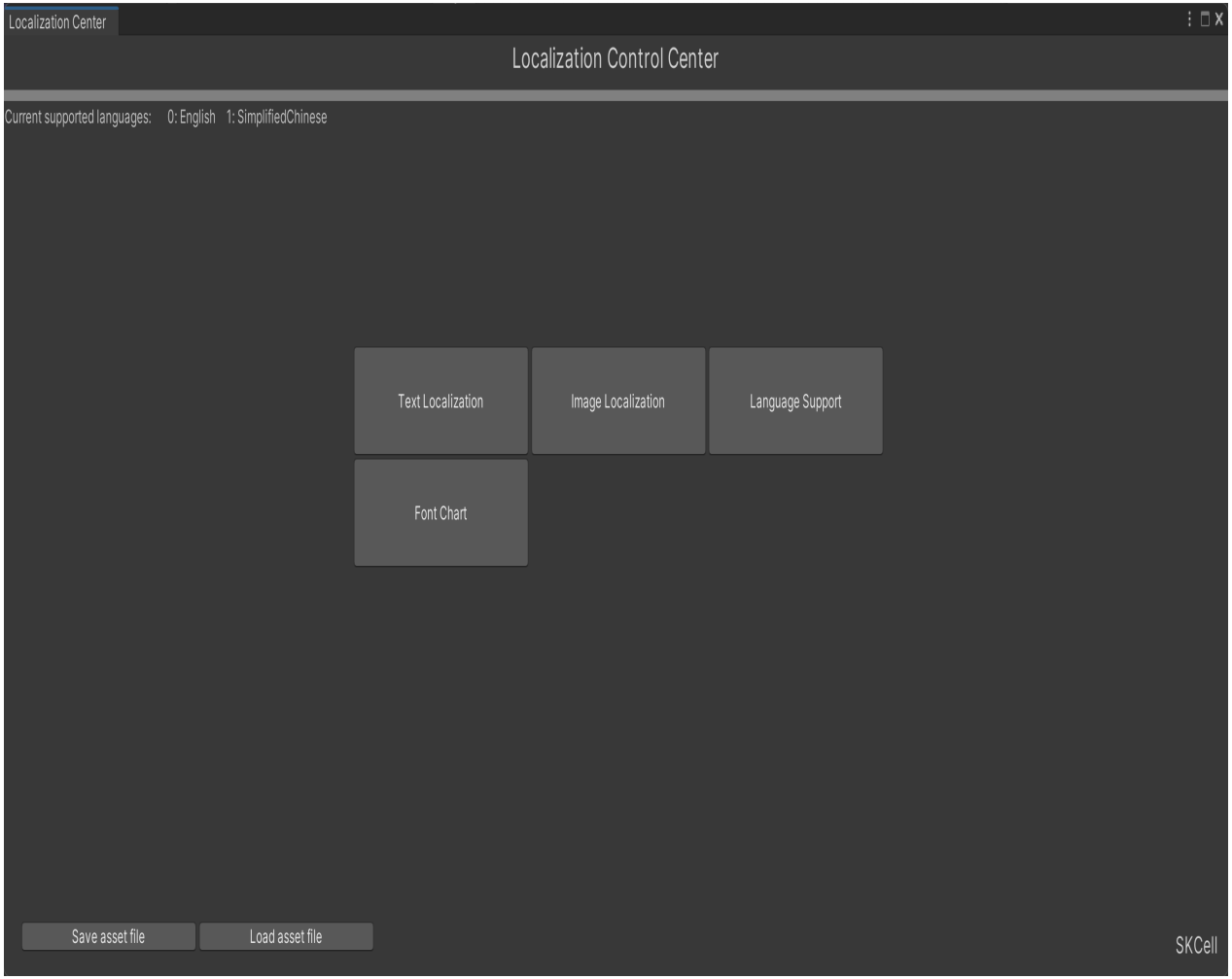
You can access a text entry by:

```csharp
SKLocalization.GetLocalizationText(int localID);
```

## 9.2 Localization Control Center

Localization Control Center is the place where you can manage and edit the localizaed content.

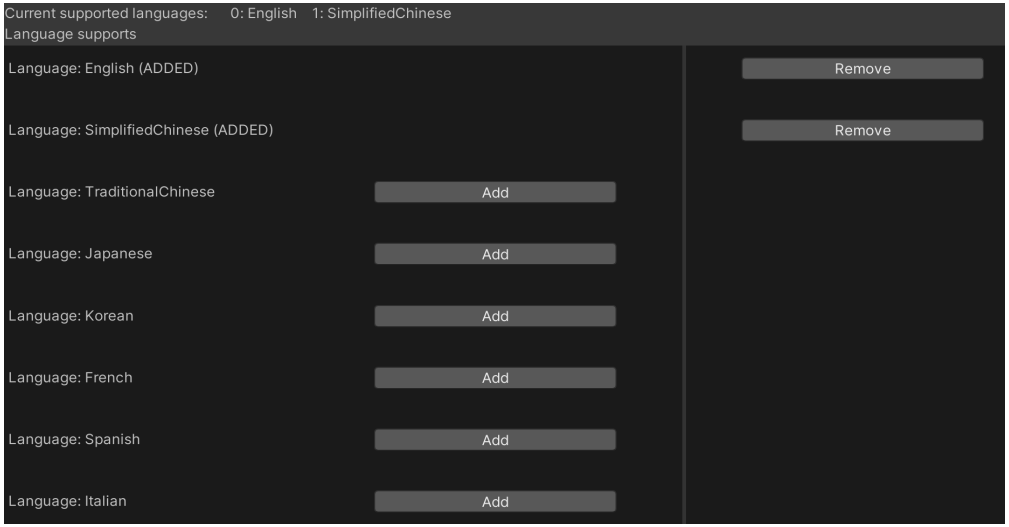
Open by selecting **Tools/SKCell/Localization Center** from the menu bar.




There are four sections: **Text Localization**, **Image Localization**, **Language Support**, and **Font Chart**.

Text and Image Localization sections simply contains all the **Local Entries**. You can add/edit/remove entries here.

### Language Support




You can add/remove supported languages here. **Only supported languages are visible in text and image localization sections.**
Each local entry contains all contents of the available languages. Therefore removing a supported language will not delete the data related with that language. If you add the language back again, the content will appear.
Supported languages are only filters by which the local entries display.

### Font Chart

You can assign different fonts for each local entry.
Do so by filling the **Font Override** field beside each local entry. Each font corresponds to an integer ID which you can specify in the font chart.

### Saving & Loading

You can save / load the localization asset by pressing the buttons at the bottom of the control center.
**There can only be one localization asset for each project.**

**Auto saving is not available!** Be sure to hit the Save button each time you finish editing!

## 9.3 Usage

Text localization is bestly integrated with the **SKText** component.

### Text Localization with No Arguments

To localize texts with no arguments (meaning that they are immutable), simply fill in the text entry with the predetermined translation for each language, note down the local ID, then use the **UpdateLocalID** method:

csharp

```
//EXAMPLE
const int MY_LOCAL_ID = 5135;
skText.UpdateLocalID(MY_LOCAL_ID);
```

Alternatively, you can fill in the local ID in the inspector as well. This will act as a initial value.

### Text Localization with Several Arguments

Oftentimes the text we want to localize is not static. For example,

ENG: There are 3 apples.
CHN: 那里有3个苹果。

Notice that there exists a variable in the sentence, (3 in this example).

In this case, you can leave formatting indicators in the sentence to reserve places for these variables. (like when doing string formatting)
Therefore you would fill in the local entries like this:

ENG: There are {0} apples.

CHN: 那里有{0}个苹果。

Then use the **UpdateText** method:

csharp

```
//First update the local id
const int MY_LOCAL_ID = 5135;
skText.UpdateLocalID(MY_LOCAL_ID);
//Then pass in the arguments
skText.UpdateText(3);
```

Similarly, this works for at most 3 arguments.

ENG: You score is {0}. Your best score is {1}. Time: {2}.
CHN: 得分：{0}；最高分：{1}；用时：{2}.

Then use the **UpdateText** method:

csharp

```
//First update the local id
const int MY_LOCAL_ID = 5135;
skText.UpdateLocalID(MY_LOCAL_ID);
//Then pass in the arguments
skText.UpdateText(14000,23000,90.95f);
```

The text will be formatted as:

ENG: You score is 14000. Your best score is 23000. Time: 90.95.

CHN: 得分：14000；最高分：23000；用时：90.95.

### Image Localization

Text localization is bestly integrated with the **SKText** component.
This works similar as text localization with no arguments.

First fill in the local entries, then use the **UpdateLocalID** method:

```csharp
//EXAMPLE
const int MY_LOCAL_ID = 5135;
skImage.UpdateLocalID(MY_LOCAL_ID);
```

## 9.4 SKLocalizationManager

SKLocalizationManager manages the localization system globally.

It provides the following helper functions:

**void Localize(GameObject go)**
*Localize a single object (works with all SKText and SKImage components on it).*

```csharp
SKLocalizationManager.Localize(go);
```

**void LocalizeAll(LanguageSupport language)**
*Localize all objects in the scene with a certain language.*

csharp

```
SKLocalizationManager.LocalizeAll(LanguageSupport.French);
```

**void LocalizeAll()**
*Localize all objects in the scene according to the language setting in **SKEnvironment**.*

csharp

```
SKEnvironment.curLanguage = LanguageSupport.French;
SKLocalizationManager.LocalizeAll();
```

**void LocalizeAllChildren(GameObject go)**
*Localize all (recursive) children of a game object according to the language setting in **SKEnvironment**.*

# 10. Grid System

SKCell offers a 2D grid system with pathfinding.

## 10.1 SKGrid

SKGrid is the basic data class for a grid.

A grid is an (m x n) 2D tilemap with square tiles starting from the **bottom left**. Each cell has a cell ID (x, y) that represents its position in the grid. (cell ID starts from 0)

These are the basic info in the SKGrid class:

```csharp
public float cellSize;
public int width, height;
```

Each cell can also hold a float value:

```csharp
public float[,] cellValues;
```

### Methods

**Vector2Int PositionToCell(Vector3 pos)**
*World position to cell ID.*

**Vector3 CellToPosition(Vector2Int cell)**
*Cell ID to world position.*

**Vector3 GetCellCenterPos(int x, int y)**
*Cell ID to center position (world).*

**Vector3 GetCellBottomLeftPos(int x, int y)**
*Cell ID to bottom left position (world).*

**Vector3 GetGridCenterPos()**
*Get center position (world) of the entire grid.*

**Vector2 GetMousePosCell(Camera cam)**
*Get the cell over which the mouse is hovering.*

**void SetCellValue(int x, int y, float value)**
*Set the value for a cell.*

**float GetCellValue(int x, int y)**
*Get the value of a cell.*

**int CellDistance(Vector2Int c1, Vector2Int c2)**
*Get the Manhattan distance between two cells.*

**float CellDistanceCir(Vector2Int c1, Vector2Int c2)**
*Get the Euclidian distance between two cells.*

There are also fields and methods for pathfinding.

## 10.2 SKGridLayer

SKGridLayer is the monobehaviour component that contains an SKGrid.

You can see the grid in scene view with **Gizmos** turned on.

### Structure

**Generate New Grid**
Generate a new grid. **Will replace the existing grid.**

**Apply Changes**
If you have changed the width, height, size, or resolution settings of the grid, click this to apply those changes.

**Save/Load Grid**
Save to / load from the grid asset.

**Grid Asset**
Each grid asset contains an SKGrid with all its settings. To create a new asset, click **New**.

**UID**
Unique identifier for this grid component.

**Draw Text**
Visualize cell IDs.

**Width/height/cell size**
Self-explanatory. Besure to **Apply Changes** after modifying these values.


## Display

Each grid layer has a underlying texture2D for display.

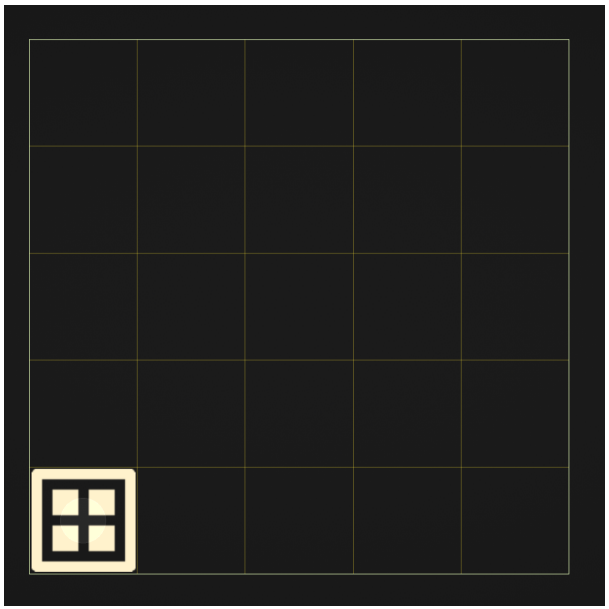
**Resolution**
Resolution of the display texture.
**Default Color**
Default color of the display texture.
**Filter Mode**
Filter mode of the display texture.

**Pathfinding**
See **Pathfinding**.

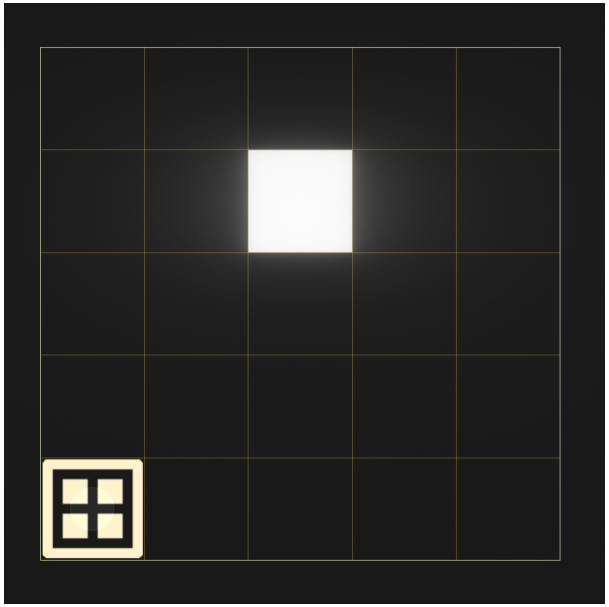
### Editing the Grid

You can edit the grid in scene view by selecting **Edit Grid** in the inspector.

A grid gizmo will appear:

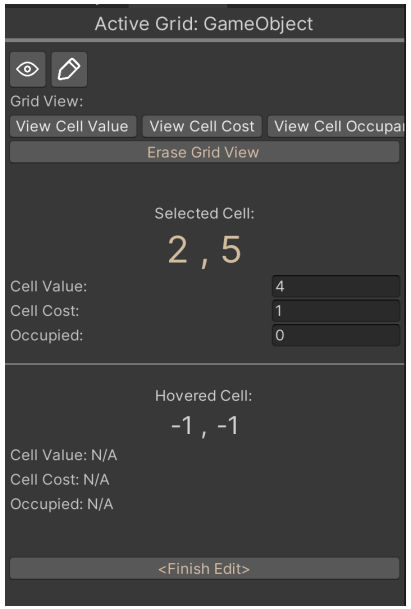


You can select a cell to check its info by right-clicking on it:



On the left of the screen, a grid editor will appear where you can check and edit the cell infos.

Be sure to click **Finish Edit** after editing.



## Methods

**Vector2Int CellFromWorldPos(Vector3 pos)**

*World position to cell ID.*

**Vector3 WorldPosFromCell(Vector2Int cell)**

*Cell ID to world position.*

**List CellsFromWorldPos(Vector3 pos1, Vector3 pos2)**

*Get all cells covered in the square area defined by pos1 and pos2.*

**List CellsFromCoordinate(Vector2Int cell1, Vector2Int cell2)**

*Get all cells in the range defined by two cell IDs.*

**void SetCellColor(int x, int y, Color color, bool apply = true)**

*Set the color of a cell.*

*apply: if true, the changes will be visible immediately.*

**void SetCellColor(List cells, Color color, bool apply = true)**

*Set the color of multiple cells.*

**void EraseCellColor(int x, int y, bool apply = true)**

*Reset a cell color to the default.*

**void EraseCellColor(List cells, bool apply = true)**

*Reset multiple cell colors to the default.*

**void AddCellColor(int x, int y, Color colorInc, bool apply = true)**

*Add to a cell color.*

**Color GetCellColor(int x, int y)**

*Get a cell color.*

**void SetCellColorRange_Circle(CellOperator op, int x, int y, Color c1, Color c2, int radius)**

*Set cell colors in a circular range.*

*op: set, add, or multiply.*

*c1: color at the center of the range.*

*c2: color at the border of the range.*

**void SetCellColorRange_Square(CellOperator op, int x, int y, Color c1, Color c2, int radius)**

*Set cell colors in a square range.*

*op: set, add, or multiply.*

*c1: color at the center of the range.*

*c2: color at the border of the range.*

**void SetCellColorRange_Star(CellOperator op, int x, int y, Color c1, Color c2, int radius)**

*Set cell colors in a star range.*

*op: set, add, or multiply.*

*c1: color at the center of the range.*

*c2: color at the border of the range.*

**void SaveGridToAssets(SKGridAsset asset)**

*Save the current grid to the grid asset.*

**void LoadGridFromAssets(SKGridAsset asset, bool generateStructure = true)**

*Load the current grid from the grid asset.*

## 10.3 Pathfinding

SKGrid supports three pathfinding algorithms:
A*, B*, and BFS.

The grid holds the following values for each cell:

csharp

```
public int[,] pf_ParentCell; //0:Left, 1:Right, 2:Up, 3:Down, 4:LU, 5:LD, 6
public float[,] pf_FValue; //F(x)=G(x)+H(x)
public int[,] pf_CellCost; //0:Free to walk, 1:Unwalkable
```

### Using Pathfinding

Using pathfinding to generate a path, simply call: **SKGridLayer.PathfindingStart(Vector2Int startCell, Vector2Int endCell)**. This returns a list of cells that represents the shortest path.

csharp

```
//Get a path from (1,2) to (5,9)
SKGridLayer grid;
List<Vector2Int> myPath = grid.PathfindingStart(new Vector2Int(1,2), new Ve
```

Alternatively, you can also do this separately:

```csharp
//Get a path from (1,2) to (5,9)
SKGridLayer grid;
grid.PathfindingSetStartPoint(1,2);
grid.PathfindingSetEndPoint(5,9);
List<Vector2Int> myPath = grid.PathfindingStart();
```

### Setting Obstacles

To assign walkability for each cell, set their **cell values**. A cell value of 0 means free to walk, while a cell value of 1 means unwalkable.
Run pathfinding then, the algorithm will give you a shortest path without using unwalkable cells.

You can also ignore the cell costs using this overload:

```csharp
List<Vector2Int> PathfindingStart(bool ignoreCellCost = false)
```

### Pathfinding Directions

There are two direction settings available:

- 4 directions (up, down, left, right)
- 8 directions (up, down, left, right, up-left, up-right, down-left, down-right)

This defines which cells we can go when doing pathfinding. You can set this in the inspector or by setting

```csharp
SKGridLayer.directionAllowed
```

### 10.4 Grid Occupancy

*[Legacy] Usable but can be replaced by cell values.*

## 11. Path Designer

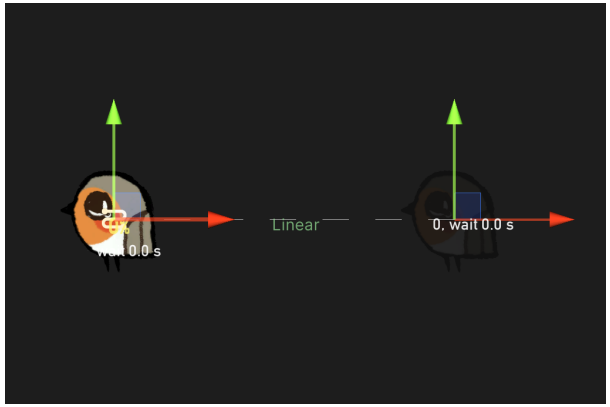
The path designer component gives an easy solution to make an object move along a path. This can be used to create moving platforms, enemies, simple animations, etc.

### 11.1 Getting Started

To use SKPathDesigner,

1. Turn on **Gizmos**.
2. Create a sprite.
3. Add SKPathDesigner component onto it.
4. Click **Add Waypoint** in the inspector.
5. Play!

You should see your sprite move horizontally for several seconds.

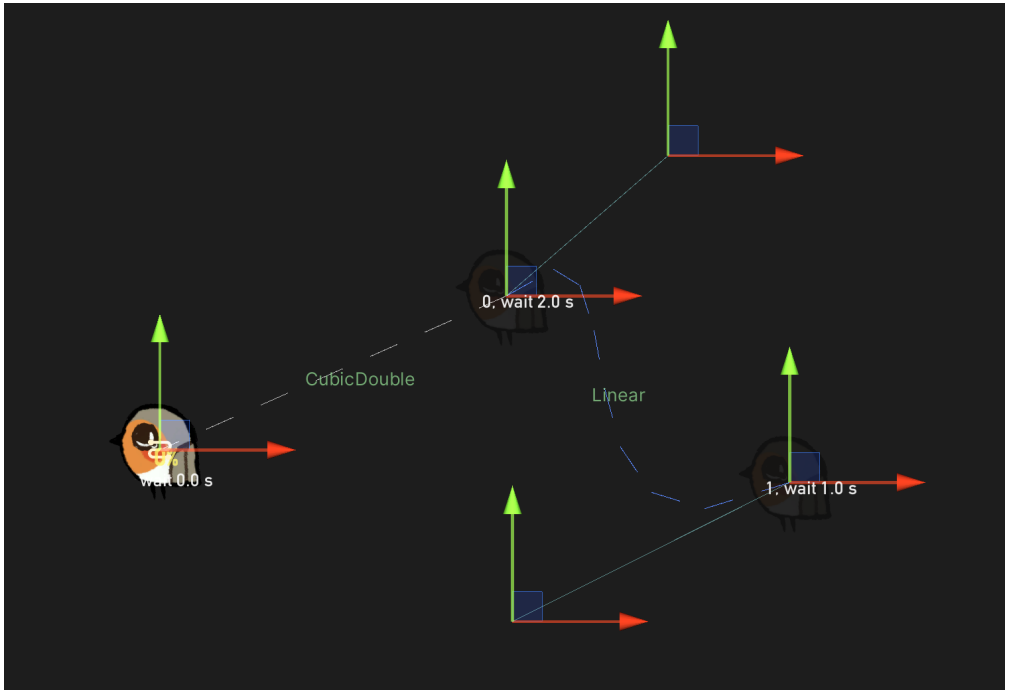



## 11.2 Waypoints

A path is comprised of multiple **waypoints**.
In SKPathDesigner, each waypoint has the following properties:

- ID
- Type (line or curve)
- Wait time (seconds to wait on this waypoint)
- Position
- Movement curve (animation curve to define how fast the object moves along the way)
- Bezier control points (if this waypoint corresponds to a curve, use 2 control points to make the curve)

For example:




In this scenario, there are three waypoints:

**Waypoint Start**: This is where the path starts.

**Waypoint 0**:
- Type: Line
- Wait time: 2.0s
- Movement curve: Cubic Double

**Waypoint 1**:
- Type: Curve
- Wait time: 1.0s
- Movement curve: Linear

### Movement Curve

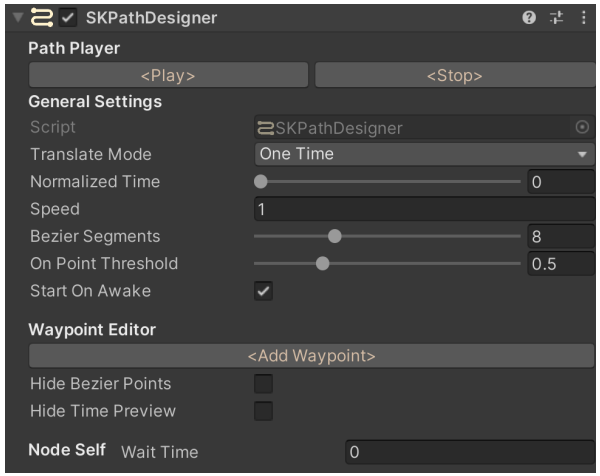
Visualize each line segment between waypoints as a segment from 0 to 1. The movement curve uses an **Animation Curve** to define its speed along the way. For example, **Cubic Double** represents a slow-fast-slow movement.

### Bezier Curves

SKPathDesigner uses **Bezier Curves** to represent curves. Each segment requires 2 control points each extending in tangent direction from a waypoint.

## 11.3 SKPathDesigner

### Structure




**Path Player**
This allows you to preview the movement along the path in scene view.

**Translate Mode**
There are 3 modes available:
- One time
- Ping pong
- Repeat

**Normalized Time**
Drag this to see the movement along the path. 0 is starting point, 1 is ending point.

**Speed**
Speed of movement.

**Bezier Segments**
Smoothness of bezier curves. More segments means smoother curves and lower efficiency.

**On Point Threshold**
How close should the object be from
waypoints to be considered "arrived"?

**Start on Awake**
Play the sequence on awake.

**Waypoint Editor**
Add, delete, and modify waypoints here.

### Methods

**void StartPath()**
*Start movement along this path. Will restart if called during a movement.*

**void PausePath()**
*Pause the ongoing movement.*

**void AddWaypoint(SKTranslatorWaypoint waypoint)**
*Add a new waypoint.*

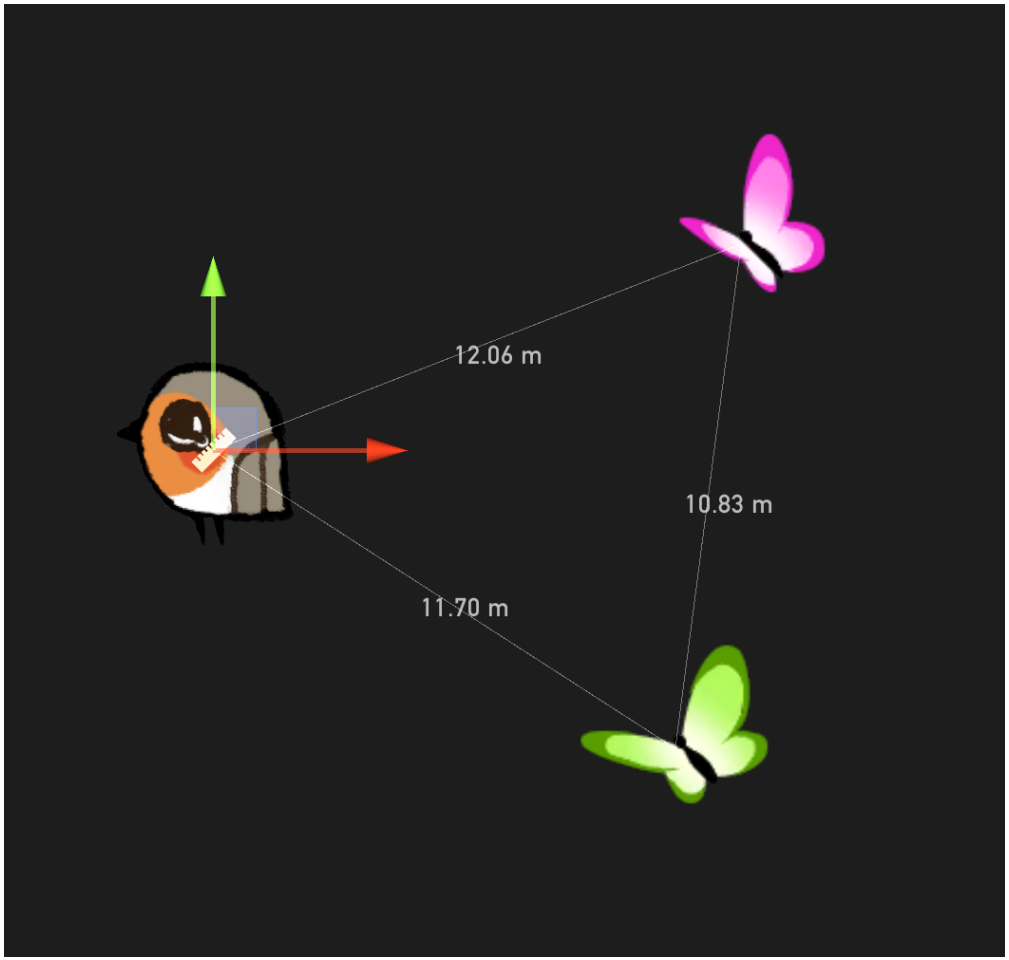
**void DeleteWaypoint(SKTranslatorWaypoint waypoint)**
*Remove a waypoint.*

**Vector3 GetNormalizedWPosition(float t)**
*Get position along the path. t from 0 to 1 representing the whole path.*

# 12. Physics & Movement

## 12.1 SKMeasurer

SKMeasurer is a simple editor tool to display distances between objects.




### How to Use

1. Select the object from which you want to measure.
2. Add the destination objects in the inspector.

### Measure Modes

There are two measure modes:
- **Every Object.** In this mode, distances of all pairs of measured objects will be displayed.
- **Next Object.** In this mode, only the distances of consecutive pairs of measured objects will be displayed.

## 12.2 SKSceneObjectResponder

SKSceneObjectResponder is a component to recieve and send scene object mouse events:

- OnMouseOver
- OnMouseExit
- OnMouseDown
- OnMouseUp
- OnMouseEnter
- OnMouseDrag

### How to Use

1. Add a collider and an SKSceneObjectResponder to your object.
2. Register to the events:

```csharp
SKSceneObjectResponder responder = go.GetComponent<SKSceneObjectResponder>(
responder.onMouseOver += MyOnMouseOver;
```

You can also get the state of mouse over by:

csharp

```
responder.isMouseOver
```

## 12.3 SKColliderResponder

SKColliderResponder is a component to recieve and send collision and trigger events:

- OnTriggerEnter
- OnTriggerStay
- OnTriggerExit
- OnCollisionEnter
- OnCollisionStay
- OnCollisionExit
- OnTriggerEnter2D
- OnTriggerStay2D
- OnTriggerExit2D
- OnCollisionEnter2D
- OnCollisionStay2D
- OnCollisionExit2D

### How to Use

1. Add a collider, a rigidbody and an SKColliderResponder to your object.
2. Select which type of event you want to enable:
    - Trigger3D
    - Collision3D

- Trigger2D
- Collision2D

3. Register to the events:

```csharp
SKColliderResponder responder = go.GetComponent<SKColliderResponder>();
responder.OnTriggerEnter += MyOnMouseOver;
```

You can also set the **max count of events sent** by setting in the inspector. (e.g. you only want to use this collider once)

```csharp
responder.maxEventCount
```

A max event count of **-1** means unlimited event count.

## 12.4 SKCldResponderManager

SKCldResponderManager gives a way to manager all the SKColliderResponders in the scene.

Each responder can be given a string ID in the inspector.
On start, responders will register themselves to the responder manager.

### Methods

**SKColliderResponder GetResponder(string uid)**
*Get a responder by its uid.*

**SKColliderResponder GetLastResponder()**
*Get the last responder which invokes a valid event.*

In this way, you don't need references to get collider events.

csharp

```
//EXAMPLE: Add event when the head collider is hit
string CLD_TAG_HEAD = "cld_head";
SKColliderResponder head_res = SKCldResponderManager.GetResponder(CLD_TAG_H
head_res.OnCollisionEnter += OnHeadHit;
```

## 12.5 SKObjectDragger

Add SKObjectDragger to an object, you can drag it with the mouse.

### How to Use

1. Add a collider and an SKObjectDragger to an object.
2. Play and drag the object!

### Translation Constraints

You can enable/disable translation on the X and Y axes by setting in the inspector.

### Inertia

You can enable/disable/edit inertia in the inspector. This allows the object to move a little further after you stop dragging.

### Events

The following events are available:
- OnBeginDrag
- OnDrag
- OnEndDrag
- OnStop

## 12.6 SKObjectRotater

Add SKObjectRotater to an object, you can rotate it with the mouse.

### How to Use

1. Add a collider and an SKObjectRotater to an object.
2. Play and rotate the object!

### Sensitivity

You can adjust the mouse sensitivity in the inspector.

### Rotation Constraints

You can enable/disable rotation on the X and Y axes by setting in the inspector.

### Inertia

You can enable/disable/edit inertia in the inspector. This allows the object to rotate a little more after you stop dragging.

#### Events

The following events are available:
- OnBeginDrag
- OnDrag
- OnEndDrag
- OnStop

## 12.7 SKObjectScaler

Add SKObjectScaler to an object, you can scale it with the mouse.

#### How to Use

1. Add a collider and an SKObjectScaler to an object.
2. Play and scale the object!

#### Sensitivity

You can adjust the mouse sensitivity in the inspector.

#### Scaling Constraints

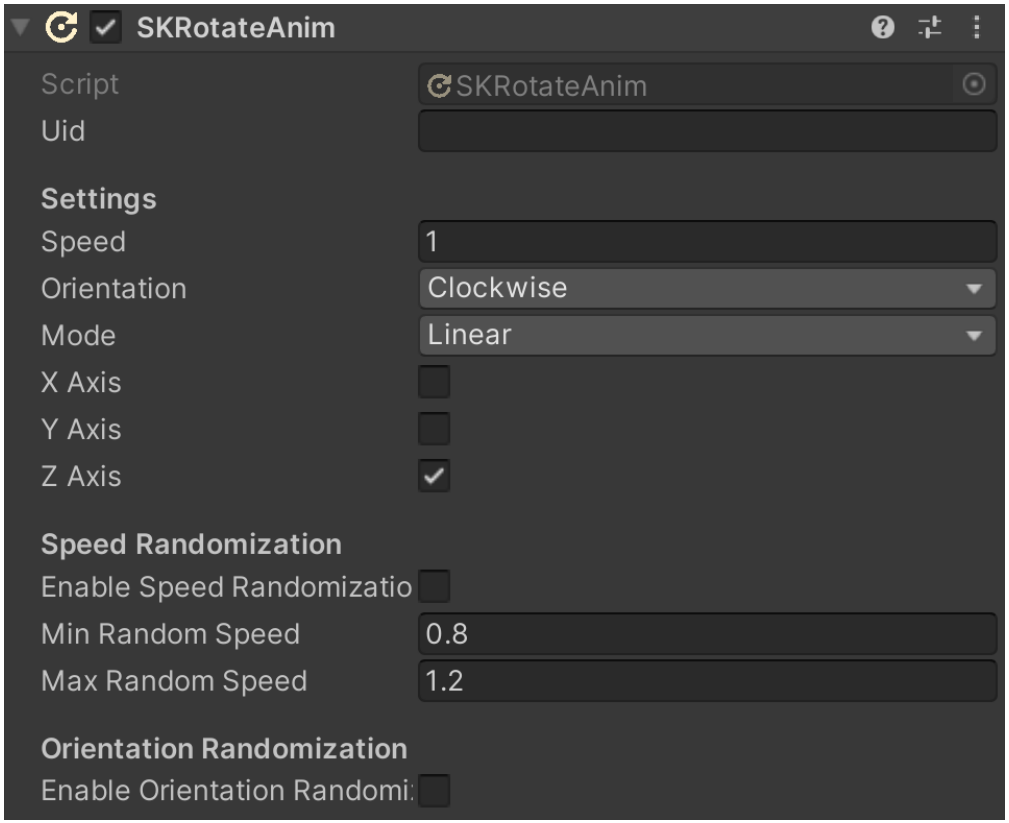
You can enable/disable scaling on the X, Y, and Z axes by setting in the inspector. There are also constraints on the min/max scales.

#### Events

The following events are available:
- OnBeginDrag
- OnDrag
- OnEndDrag
- OnStop

## 12.8 SKRotateAnim

SKRotateAnim lets you add simple rotation animation to an object.




**Speed**
*Speed of rotation.*

**Orientation**

*Clockwise or counterclockwise.*

**Mode**
*Linear or ping-pong.*

**Axes**
*Rotate around the X, Y, or Z axis.*

**Speed Randomization**
*Randomize rotate speed on start.*

**Orientation Randomization**
*Randomize orientation on start.*

#### How to Use

1. Attach SKRotateAnim to an object.
2. Play!

## 13. Audio & Video

With SKCell, you can easily deploy audio and video clips in your game.

### 13.1 SKAudioManager

#### How to Use

1. Under the **Assets** directory, create two folders: **Assets/Resources/AudioClip/Sound** and **Assets/Resources/AudioClip/Music**.
2. Put your audio clips in these folders.
3. Create a new game object and add to it the SKAudioManager component.
3. Call

```csharp
SKAudioManager.PlaySound(...);
```

or

```csharp
SKAudioManager.PlayMusic(...);
```

to play the audio clips!

*There are no real difference between sound and music. They are just there as an indentifier.*

### Methods

**AudioSource PlaySound(string id, Action action = null, bool loop = false, float volume = 1f, float pitch = 1f, float damp =0.5f)**
*Play a sound.*
*action: the action to call after playing the sound.*

```csharp
//EXAMPLE: you have clip.wav under Assets/Resources/AudioClip/Sound
audioManager.PlaySound("clip"); //Play the clip
audioManager.PlaySound("clip", null, true); //Loop the clip
```

**AudioSource PlayIdentifiableSound(string fileName, string id, bool loop = false, float volume = 1, float damp = 0.5f)**

*Play a sound and assign to it an ID (for stopping it).*

```csharp
//EXAMPLE: you have clip.wav under Assets/Resources/AudioClip/Sound
audioManager.PlayIdentifiableSound("clip","id_001"); //Play the clip
audioManager.StopIdentifiableSound("id_001"); //Stop the clip
```

**void StopIdentifiableSound(string id, float dampTime = 0.15f)**

*Stop a sound according to an ID.*

**void StopSound()**

*Stop the last sound.*

**AudioSource PlayMusic(string id, bool loop = true, int type = 2, float volume = 1f)**

*Play a music.*

**void ChangeSoundVolume(float volume)**

*Change the overall volume of sound.*

**void ChangeMusicVolume(float volume)**

*Change the overall volume of music.*

## 13.2 SKVideoManager

SKVideoManager allows you to play video clips by one line of code.

### Play a video clip fullscreen

Use the **VideoPlayer PlayVideo(string path, bool isLoop=false)** method.
*path: this is the resources path.*

```csharp
SKVideoManager.PlayVideo("Opening/FirstScene");
```

This method plays the video fullscreen.

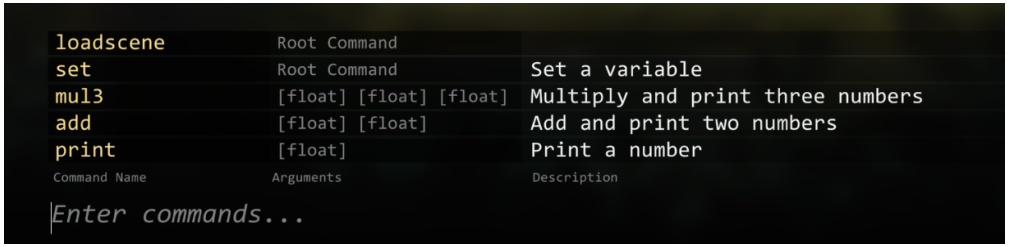
### Play a video clip on UI component

You can play video clips on a **RawImage**.
Use the **void PlayVideoInUI(string path, RawImage rawImage)** method.
*path: this is the resources path.*

```csharp
RawImage screen;
SKVideoManager.PlayVideo("Opening/FirstScene", screen);
```

## 14. In-game Console

SKCell offers a powerful and fully extendable in-game console system.

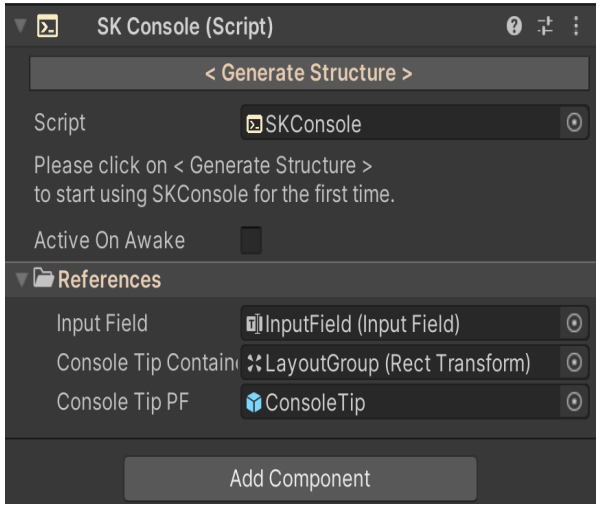



## 14.1 Getting Started

**To setup SKConsole for your project:**

1. Create a new game object.
2. Attach the **SKConsole** component to it.
3. Click the **Generate Structure** button.

Now you have it!




## 14.2 Open/Close the Console

- You can use the following methods to open/close the console once setup.

csharp

```
SKConsole.Open();
SKConsole.Close();
SKConsole.Toggle(); //Open cons
```

## 14.3 Adding and Executing Commands

**Adding a direct command to the console**
- To add a direct command (that is, in the top-level hierarchy), do this:

csharp

```
//Add a command that does nothing. (can be used to build hierarc
SKConsole.AddCommand("CmdName", "Description");

//Add a command that executes an action with no parameters.
SKConsole.AddCommand("CmdName", "Description", ()=>{
    //...
});

//Add a command that executes an action with 1 float parameter.
SKConsole.AddCommand("CmdName", "Description", (x)=>{
    //...
});

//Add a command that executes an action with 2 float parameters.
```

```
SKConsole.AddCommand("CmdName", "Description", (x,y)=>{
    //...
});

//Add a command that executes an action with 3 float parameters.
SKConsole.AddCommand("CmdName", "Description", (x,y,z)=>{
    //...
});
```

### Adding sub-commands to the console

- To add a sub-command (that is, not in the top-level hierarchy), do this:

```csharp
//Add a root command that does nothing. (serves as a menu)
SKConsoleCommand cmd = SKConsole.AddCommand("RootCmdName", "D
//Add a sub-command that executes an action.
cmd.AddCommand("SubCmdName", "Description", ()=>{
//...
})
```

We can then execute this sub-command by typing **RootCmdName SubCmdName** in the console.

### Executing a command

- To execute a direct command, simply type out the command in the console and **hit enter**.

### 14.4 Example Scene

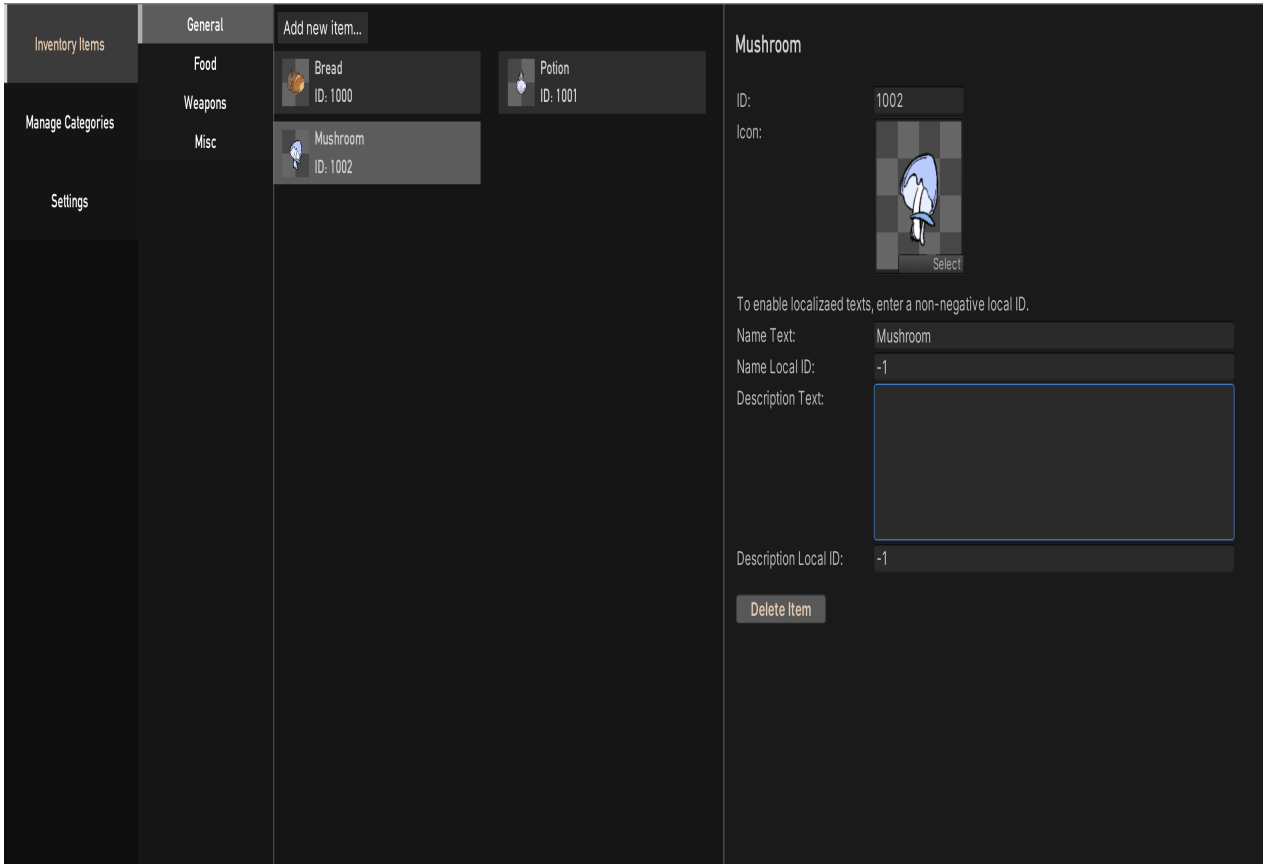
**You can see more examples in the SKConsoleScene.**

## 15. Inventory System

SKCell offers an inventory system with an editor database and customizable UI panels.

### 15.1 SK Inventory Center

SK Inventory Center is a editor-based database where you can build the item data for your project. To open the inventory center, select **Tools/SKCell/Inventory Center** from the editor menu dropdown.




There are three sections: **Inventory Items**, **Manage Categories**, and **Settings**.

**Inventory Items**

This is the place to add, delete, and edit item data.

Each item belongs to a **category**, as shown on the left side. To *add an item* to the selected category, press the **Add new item...** button on the top of the section.

Each inventory item has the following attributes:

**ID:** The identifier of the item (used to index the item in a dictionary). Note that duplicate item IDs are not allowed: an error will be thrown for any duplicate item IDs.

**Icon:** The icon (Texture2D) of the item.
**Name Text:** The name of the item. *Use this if you DO NOT want to have localization for this text.*
**Name Local ID:** The local ID of the item name. *Use this if you DO want to have localization for this text.*
**Description Text:** The description of the item. *Use this if you DO NOT want to have localization for this text.*
**Description Local ID:** The local ID of the item description. *Use this if you DO want to have localization for this text.*

**Categories**
This is the place to add, delete, and edit item categories.

Each category has the following attributes:
**Is Active:** If true, the category will be displayed in the **Inventory Items** section.
**Name Text:** The name of the category. *Use this if you DO NOT want to have localization for this text.*
**Name Local ID:** The local ID of the category name. *Use this if you DO want to have localization for this text.*

**Settings**
This is the place to set preferences for the SK Inventory Center.

## 15.2 SKInventoryItemData

SKInventoryItemData is the class holding the data for an inventory item. Its definition is as follows:

```csharp
public class SKInventoryItemData
{
    public int id;

    public int name_LocalID, descrip_LocalID;
    public string name, description;

    public Texture2D icon;
    public int category;

    public Vector2Int slotSize = Vector2Int.one;

    public bool canUse;
    public Action onUse;
}
```

*The canUse and onUse fields cannot be edited directly from the inventory center. For examples of their usage, please refer to section 15.5.*

## 15.3 SKInventoryItem

SKInventoryItem is the class representing an item in an SKInventory. Its definition is simple:

```csharp
public class SKInventoryItem
{
    public int id;
```

```
        public int count;
    }
```

## 15.4 SKInventory

SKInventory is the class representing one inventory. It also offers some useful static methods.

### Public Methods

**AddItem(int id, int count, bool stacking = true)**
*Add an item to this inventory.*
***id**: Item id as specified in the SK Inventory Center.*
***count**: Item count.*
***stacking**: If enabled, items with the same id will stack.*

**void RemoveItem(int id, int count)**
*Remove a certain count of an item from this inventory.*

**void RemoveItemDirectly(int id)**
*Remove an item from this inventory, regardless of its count.*

**bool ContainsItem(int id)**
*Does this inventory contain the given item?*

**void RemoveItemDirectly(int id)**
*Remove an item from this inventory, regardless of its count.*

**SKInventoryItem GetItem(int id)**
*Get an item from this inventory.*

**void Clear()**
*Remove all items from this inventory.*

### Static Methods

**static SKInventoryItemData GetItemData(int id)**
*Get item data (from the database) by its id.*

### Examples

csharp

```
//Create a new inventory
SKInventory inventory = new SKInventory();

//Add item to the inventory (id = 1000, count = 4)
inventory.AddItem(1000, 4);

//Add item to the inventory (id = 2001, count = 12)
inventory.AddItem(2001, 12);

//Remove all items (id=1000) in the inventory.
inventory.RemoveItemDirectly(1000);
```

## 15.5 SKInventoryLayer

SKInventoryLayer is the component that provides a fully-functional UI panel that displays an SKInventory.
Each SKInventoryLayer holds ONE SKInventory object to display.


### Getting Started

To setup an SKInventoryLayer:

1. Create a new game object and add the SKInventoryLayer component.
2. Click the **Generate Structure** button.
3. The SKInventoryLayer is set up!

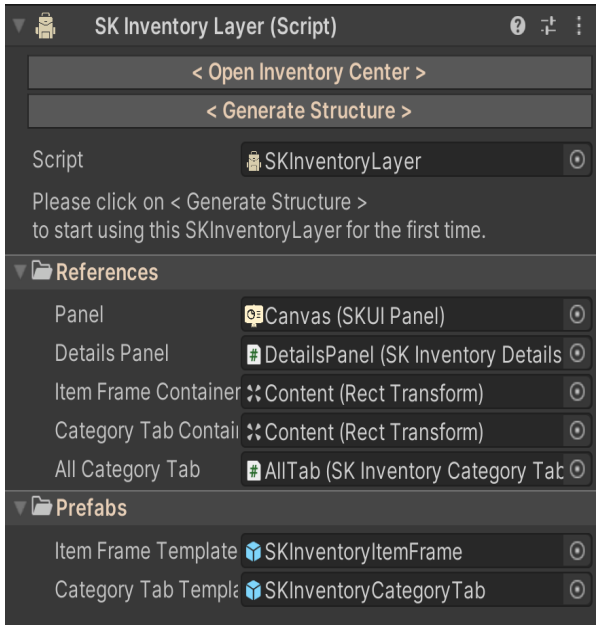
### The Component

- **References**

  *These are references to some of the objects in the inventory structure. Do not edit these unless you fully understand how they work.*

- **Prefabs**

  *SKInventoryLayer instantiates item frames and category tabs dynamically. Edit these if you want to have your own customized elements displayed instead of the default ones.*




## Properties

**SKInventory Inventory**
*The inventory object that this layer is displaying.*

## Methods

**void OpenInventoryPanel()**
*Display the UI panel of this inventory layer.*

**void CloseInventoryPanel()**
*Hide the UI panel of this inventory layer.*

**void ToggleInventoryPanel()**
*Toggle the UI panel of this inventory layer.*

### Examples

```csharp
public class SKInventory_Demo : MonoBehaviour
{
    public SKInventoryLayer inventoryLayer; //This is the component that d
    void Start()
    {
        SKInventory inventory = inventoryLayer.Inventory; //Get the invent
        inventory.AddItem(1000, 1); //Add item (id = 1000, count = 1)
        inventory.AddItem(1001, 5);//Add item (id = 1001, count = 5)

        SKInventory.GetItemData(1000).canUse = true; //Set item 1000 to be
        SKInventory.GetItemData(1000).onUse += () => { print("Use Item 100

        inventoryLayer.OpenInventoryPanel(); //Open the inventory UI panel
    }

}
```

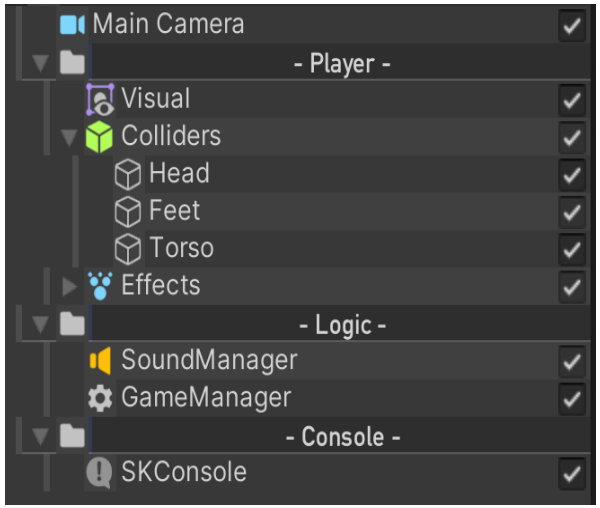
## 15.6 Example Scene

**You can see example code and scene setup in the SKInventoryScene.**

# 16. Editor Interfaces

SKCell brings a series of editor interface improvements to Unity.

## 16.1 Hierarchy Window




**Features**

- **Separators** (marked as blue)
*End a game object name with "-" to mark it as a separator.*

- **Shortcut: Enabling/Disabling** (check box to the right)
*Enable or disable a game object by selecting the check box.*

- **Customizable Icons**
*Change the icon for a game object by clicking on the icon to the left of the object.*

- **Appearance**
*Two consecutive rows have a slight color difference.*
*Colored bar to the left to indicate parent/child layers.*

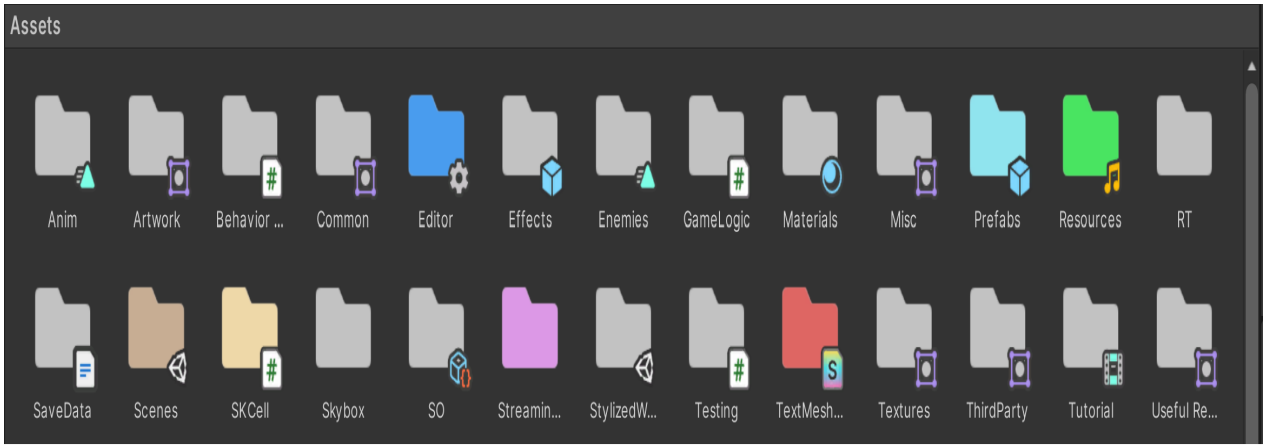
**Customizing Colors**

You can customize colors for the hierarchy window by selecting **Tools/SKCell/Hierarchy Style** from the menu bar.
Close the customization window to apply changes.

## 16.2 Project Folder Features

SKCell brings an upgrade to your project window. There will be an icon for each folder that indicates the **predominant file type** inside that folder.

The icon updates in real time.



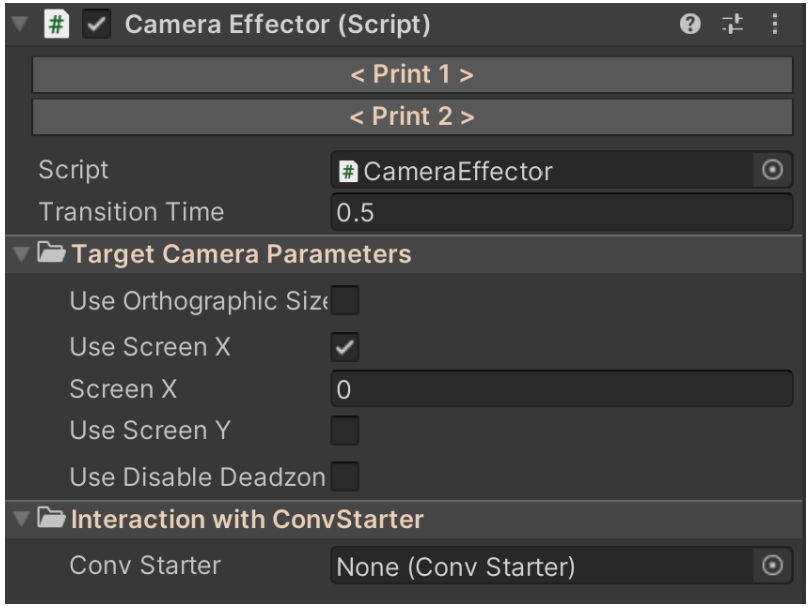

There are also special colors for designated folders.
The following code shows you the types and colors supported. (updated v0.13)

csharp

```
{"SKCell", new Color(1.0f, 0.9f,0.7f, 1f)},
{"Editor", new Color(0.3f, 0.65f,1.0f, 1f)},
{"Prefabs", new Color(0.6f, 0.95f,1.0f, 1f)},
{"Prefab", new Color(0.6f, 0.95f,1.0f, 1f)},
{"Resources", new Color(0.3f, 0.95f,0.4f, 1f)},
{"Scenes", new Color(0.83f, 0.72f,0.61f, 1f)},
{"TextMesh Pro", new Color(0.93f, 0.42f,0.41f, 1f)},
{"StreamingAssets", new Color(0.92f, 0.63f,0.96f, 1f)},
```

## 16.3 Inspector Attributes

SKCell offers a wide range of inspector attributes that allows you to make custom inspector for your scripts easily.




For example, the following code makes the inspector above.

csharp

```csharp
public float transitionTime = 0.5f;

[SKFolder("Target Camera Parameters")]
public bool useOrthographicSize;
[SKConditionalField("useOrthographicSize", true)]
public float orthographicSize;
public bool useScreenX;
[SKConditionalField("useScreenX", true)]
public float screenX;
public bool useScreenY;
```

```
[SKConditionalField("useScreenY", true)]
public float screenY;
public bool useDisableDeadzone;

[SKFolder("Interaction with ConvStarter")]
public ConvStarter convStarter;
[HideInInspector]
public CinemachineVirtualCamera cam;
[HideInInspector]
public CinemachineFramingTransposer transposer;

private float oOrthoSize, oScreenX, oScreenY;

[SKInspectorButton("Print 1")]
public void Print1()
{
    print(1);
}
[SKInspectorButton("Print 2")]
public void Print2()
{
    print(2);
}
```

### Supported Attributes

[**SKFolder**]: Makes a folder (foldout) containing the fields underneath.

csharp

```
[SKFolder("Folder Name")]
public int a;
public string b;
public GameObject c;
```

**[SKEndFolder]**: Marks the end of an SKFolder

csharp

```
[SKEndFolder]
```

**[SKInspectorButton]**: Makes a button in the inspector. Will execute a function when pressed.

csharp

```
[SKInspectorButton("Print 1")]
public void Print1()
{
    print(1);
}
```

**[SKConditionalField]**: Makes a field ONLY appear when some other conditions are met.

csharp

```
public bool useOrthographicSize;
[SKConditionalField("useOrthographicSize", true)]
```

```csharp
    public float orthographicSize;
```

**[SKFieldAlias]**: Set an alias for a field. The alias will be displayed in the inspector instead of the field name.

csharp

```csharp
    [SKFieldAlias("Player Speed")]
    public float spd;
```

**[SKSeparator]**: Makes a separator line.

csharp

```csharp
    public int a;
    [SKSeparator]
    public float b;
```

**[SKResettable]**: Makes a field resettable to its default value by clicking a reset button in the inspector.

csharp

```csharp
    [SKResettable]
    public float bounciness = 5.0f;
```

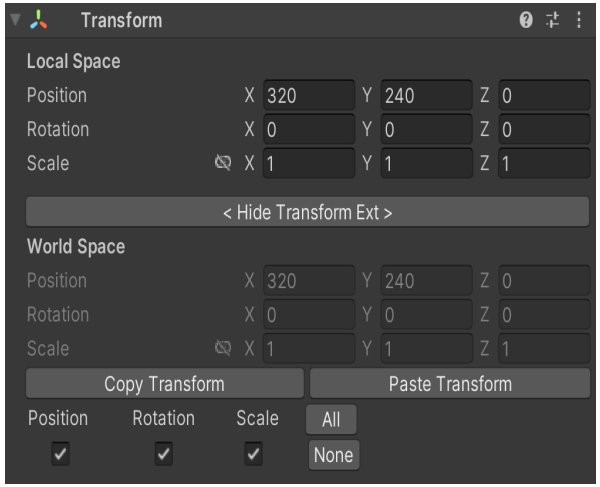
## 16.4 Transform Component

Select **Show Transform Ext** to view extra info.

**Features**

- **Local & World Spaces**
*View transform values in both local & world spaces.*

- **Copy & Paste Transform Values**
*Copy & Paste transform values between game objects.*




## 16.5 Example Scene

**You can see more examples in SKHierarchyScene and SKInspectorScene.**

# 17. Editor Tools

## 17.1 Sprite Editor

The SK Sprite Editor allows you to **draw, erase, and modify images** in the Unity Editor.


### 17.1.1 File I/O

#### Open the SK Sprite Editor

Select **Tools/SKCell/Sprite Editor** from the Unity Editor toolbar to open the editor.

#### Creating a new image

You can create a new PNG image by clicking on the **Create New Image...** box on the left.

#### Loading a file to the sprite editor

You can assign the texture you want to edit by clicking on the **Texture** box on the left.

#### Saving an edited file

You can save the changes to the CURRENT image by clicking the **Apply Changes** button on the top right corner. In this way, the original texture will be replaced.
You can also save the edited file to a new file by clicking the **Save As...** button on the left.

### 17.1.2 Sprite Editor Tools

The sprite editor inspector is located on the right of the screen. You can select tools, adjust image colors, and apply image effects to the texture.

- The **Select** tool allows you to select a rectangular area on the texture. All later modifications will be restricted to this area.
  You can also **crop the image** by clicking "Save Selected Area As..." on the left. (starting v0.15.0)

- The **Paint Brush** tool allows you to draw on the texture. You can specify a shape (smooth or solid), select a color, and adjust the size for the brush.

- The **Eraser** tool allows you to erase colors on the texture. Still, the brush settings apply.

- The **Color Picker** tool allows you to pick colors from the texture. You can also access the color picker ANYTIME by holding the right mouse button.

  **Color Adjustment**
  You can adjust the saturation, contrast, and brightness of the texture using the sliders on the right.

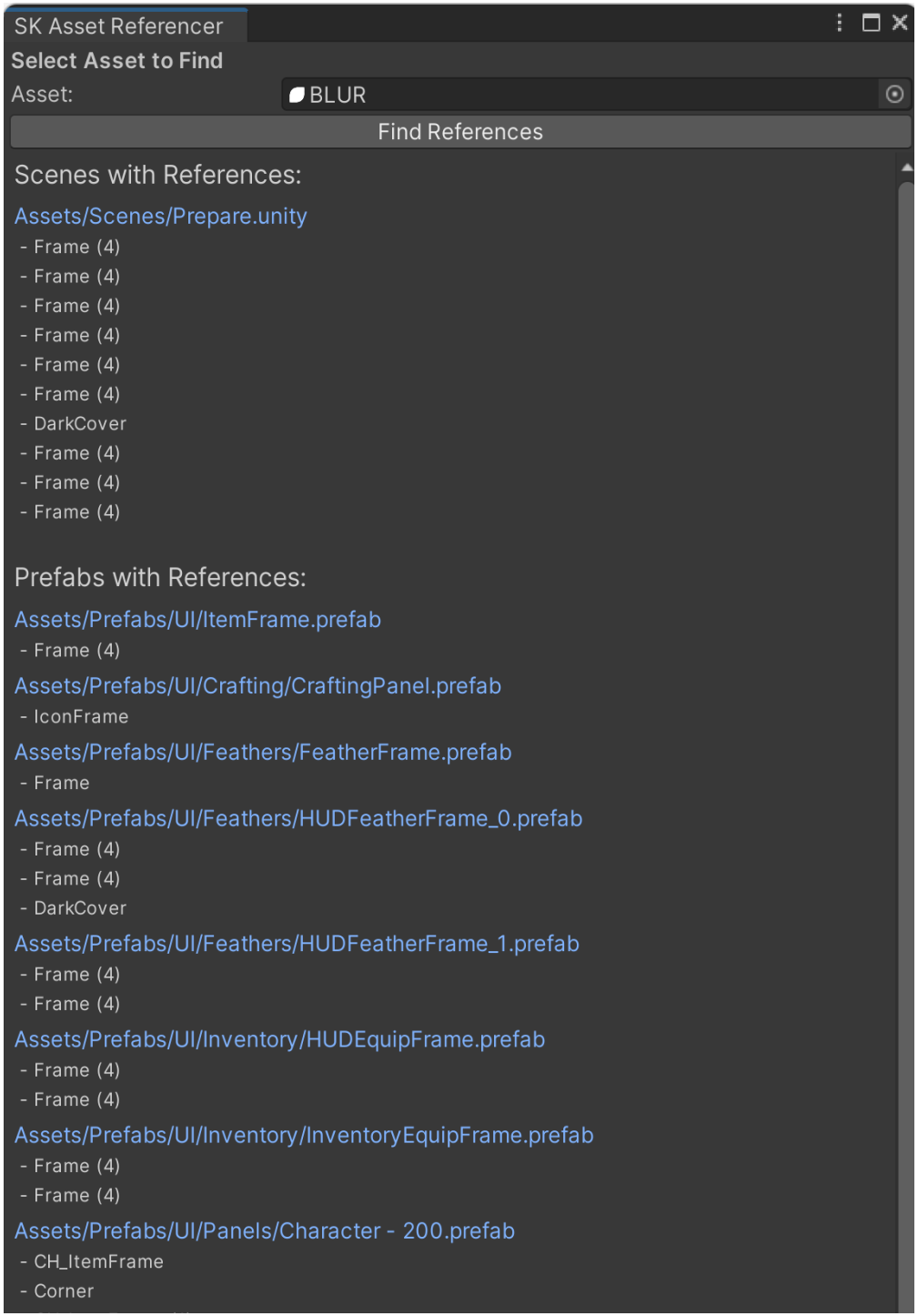
  **Erase Color**
  You can erase all colors that are within the tolerance of your specified color by clicking the Erase button on the right.

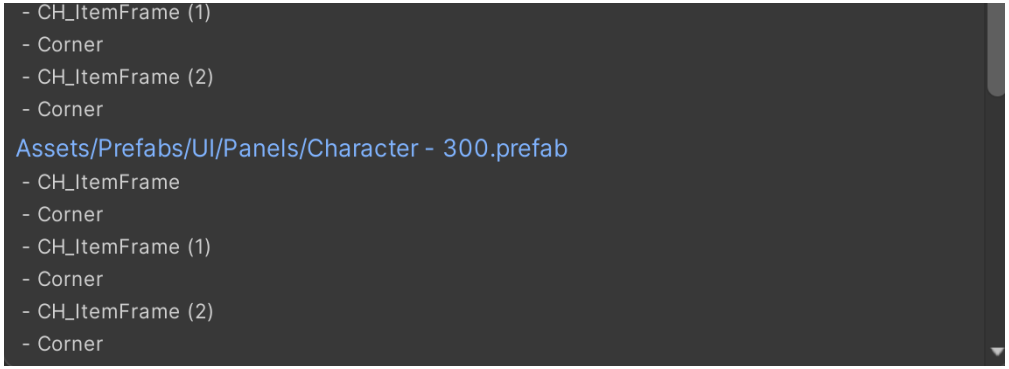
  **Gaussian Blur**
  You can apply Gaussian Blur to the image by specifying the blur radius and clicking **Blur** .

- Due to CPU limitations, this operation might take a long time for high-resolution textures.

## 17.2 Asset Referencer

The asset referencer helps you to find (if any) all references of a certain asset.


SK Asset Referencer
Select Asset to Find
Asset:
BLUR
Find References
Scenes with References:
Assets/Scenes/Prepare.unity
- Frame (4)
- Frame (4)
- Frame (4)
- Frame (4)
- Frame (4)
- Frame (4)
- DarkCover
- Frame (4)
- Frame (4)
- Frame (4)
Prefabs with References:
Assets/Prefabs/UI/ItemFrame.prefab
- Frame (4)
Assets/Prefabs/UI/Crafting/CraftingPanel.prefab
- IconFrame
Assets/Prefabs/UI/Feathers/FeatherFrame.prefab
- Frame
Assets/Prefabs/UI/Feathers/HUDFeatherFrame_0.prefab
- Frame (4)
- Frame (4)
- DarkCover
Assets/Prefabs/UI/Feathers/HUDFeatherFrame_1.prefab
- Frame (4)
- Frame (4)
Assets/Prefabs/UI/Inventory/HUDEquipFrame.prefab
- Frame (4)
- Frame (4)
Assets/Prefabs/UI/Inventory/InventoryEquipFrame.prefab
- Frame (4)
- Frame (4)
Assets/Prefabs/UI/Panels/Character - 200.prefab
- CH_ItemFrame
- Corner




To use the Asset Referencer, you can either:
1. Right-click on an asset
2. Select **SKAssetReferencer** to do the reference-check

or:
1. Select **Tools/SKCell/Tools/Asset Referencer**
2. Drag in your asset to do the reference-check

### 17.3 Sprite Colorer

The sprite colorer allows you to assign a single color to a sprite. (will affect all non-transparent pixels)
Select **Tools/SKCell/Tools/Sprite Colorer** to open the window. Detailed instructions are written there.

## Dev Log

**v1.2.0**

- Standardized all assets. (namespaces, menu paths, prefabs, file formats)
- Renamed CommonUtils to SKUtils (important!)

- Renamed MonoSingleton, Singleton to SKMonoSingleton, SKSingleton
- Brush smoothness is now correctly computed in SKSpriteEditor
- Fixed naming issues in some scripts
- Added new comments for some methods

  v1.2.1
  - Added blend mode adjustments for SKImageProcessing
  - Fixed issues with *update on play* for SKImageProcessing

    v1.2.2
  - Added SKAssetReferencer: find all references of any asset! Available by right-clicking on an asset -> SKAssetReferencer
  - Added *play on start* function to SKDialoguePlayer
  - Added *SentenceNextStep()* method to SKDialoguePlayer

    v1.2.3
  - Fixed issues with build errors
  - Minor bug fixes

    v1.2.4
  - Updated version requirement to Unity 2021.x and newer
  - Fixed issues with SKConsole_Demo.cs
  - Added more 9-sliced sprites
  - Removed unused variables

**v1.1.0**

- The Inventory update! Introducing SKInventory: a powerful inventory system with an editor database and customizable UI panels.
- Added SKInventory, SKInventoryLayer, SKInventoryItem, SKInventoryItemFrame, SKInventoryControlCenter, SKInventoryAsset
- Added ISKInventoryItemFrame, SKInventoryDetailsPanel, SKInventoryCategoryTab

- Added SKInventoryScene
- Added common sprite pack (8 sprites)
- Fixed issues with SKLocalization

  v1.1.1
  - Added localization support for SKInventory categories

    v1.1.2
  - Various bug fixes

    v1.1.3
  - Added SKTooltip and SKTooltipObject
  - Updated SKUIScene
  - Reorganized editor menus

**v1.0.0**

- Finally the first release! Available on the Unity Asset Store (in approx. 30 days).
- Adjusted project structure
- Added WorldGridXY and WorldGridXZ
- Added LocalizeAllChildren function to SKLocalizationManager
- Added new folder colors
- Removed SKModuleBase
- Removed ProjectWindowDetails
- Removed CSVProjectBuild
- Removed SKGridCellInfo
- Removed EventRef template
- Fixed issues with SKTextAnimator

  v1.0.1
  - Fixed issues with saving assets in SKLocalizationCenter
  - Removed templates in SKLocalizationCenter

- Removed some json-related functions in SKUtils

**v0.15.x**

2023.11

- The Console update!
- Added SKConsole: A powerful, easy to use, fully extendable in-game console system.
- Added SKConsoleTip, SKConsoleCommand
- Added SKUIModelViewer: View and rotate 3d model in Unity UI!
- Added *create new image* function to SKSpriteEditor
- Added *crop image* function to SKSpriteEditor
- Fixed issues with SKBehaviourEditor
- Removed Gizmos folder (unused)
- Added Consolas font

  0.15.1
  - Added start on awake function to SKConsole
  - Fixed potential problem with SKUIModelViewer
  - Added SKHierarchyScene

    0.15.2
  - Reorganized add component menus
  - Fixed issues with SKHierarchy icon loading
  - Fixed issues with SKHierarchy icon display

**v0.14.x**

2023.11

- The Shader & Effects update!
- Added effects: SKDissolveEffect, SKOuterGlowEffect, SKCartoonGrass, SKToonMaterial, SKGlitchEffect, SKWireframeEffect, SKDitherAlpha, SKEdgeOutlineEffect, SKLightCastEffect

- Added SKShaderScene
- Added SKColor, SKColorPalette
- Added SKVariableMonitor
- Fixed issues with SKTextAnimator
- Fixed issues with SKQuitControl

  0.14.1
  - Fixed display issues with SKDialogueEditor
  - Added new attribute: SKInspectorText
  - Added new project folder colors

    0.14.2
  - Added SKRandom
  - Added folder color dots for project window list-views
  - Updated SKHierarchy separator appearance
  - Removed TextureUtils

    0.14.3
  - Added SKSceneStarter
  - Added multi-edit support for most SK components
  - Fixed issues with SKHierarchy in some editor versions
  - Fixed issues with SKFolderIconSetter
  - Reorganized folder structure
  - Removed TextureUtils

**v0.13.x**

2023.10

- The Custom editor update! New features include: custom object icons, automatic project folder icons, inline inspector attributes, etc.

- Added SKFolderIconSetter, SKFolderChangeProcessor, SKAttributeDrawer, SKBehaviourEditor, SKMonoAttribute, etc.
- Added inspector attributes: SKFolder, SKEndFolder, SKConditionalField, SKResettable, SKInspectorButton, SKFieldAlias
- Enhanced SKHierarchy
- Added new editor icons

  0.13.1
  - Added project folder colors and button styles
  - Optimized project editor performance

    0.13.2
  - Added ActiveOnAwake.
  - Added more supported types for project folder colors
  - Changed hierarchy separator visuals
  - Changed SKCore inspector visuals

    0.13.3
  - Added SKUIScene for demo purposes
  - Fixed bugs associated with SKTextAnimator

    0.13.4
  - Added SKInspectorScene for demo purposes
  - Added inspector attribute: SKSeparator
  - Fixed bugs associated with SKBehaviourEditor, SKFolderIconSetter, SKSpriteProcessing, and SKImageProcessing

**v0.12.x**

2023.3

- Added SKSpriteEditor
- Added 4 tools to SKSpriteEditor: select, brush, eraser, color picker

- Added 5 utilities to SKSpriteEditor: brightness, saturation, contrast, color erase, gaussian blur
- Optimized SKHierarchy, updated appearance

  0.12.1
  - Various bug fixes

    0.12.2
  - Added SKLineAnim to Effects module

    0.12.3
  - Fixed issues with SKDialogue Scene
  - Add SKLineAnim prefab
  - Updated SKQuitControl
  - Bug fixes and routine maintenance

**v0.11.x**

2022.12

- Reorganized project
- Added full documentation
- Removed all third-party assets
- Removed ReplaceableMonoSingleton
- Removed MeshToPrefabUtils
- Removed Native Utilities
- Bug fixes

  0.11.1
  - Fixed issues with SKColliderResponder
  - Added return value of Tweening functions
  - Added API for SKPathDesigner

**v0.10.x**

2022.9

- Added Dialogue Module (SKDialoguePlayer, SKDialogueAsset, SKDialogueEditor, SKDialogueEditorNode, SKDialogueManager, SKDialoguePlayerEditor)
- Incorporated dialogue system with SKTextAnimator and SKLocalization systems
- Added fast-forward function to SKTextAnimator
- Added new environment effect packs

  0.10.1
  - Added SKQuitControl to Editor module

    0.10.2
  - Added noise texture packs
  - Bug fixes

**v0.9.x**

2022.8

- Added Movement Module (SKPathDesigner, SKPathDesignerEditor, SKPathDesignerPreview)
- Added SKBezier to Structure Module
- Added SKUIShadow, SKBackBlur to Effects Module
- Minor changes to SKCurve and SKAssetLibrary
- Added SKHierarchy to Editor Module
- Updated various icons

  V0.9.1
  - Added SKTextAnimator, SKTextAnimation, SKTextData, SKTextEffects, SKTextUtils to Effects Module
  - Added scaling, translation, rotation, wave, shake, color, and alpha effects to SKTextAnimation

V0.9.2

- Added dangle, banner, exclaim, and twinkle effects to SKTextAnimation
- Completed SKTextAnimator functionalities

V0.9.21

- Integrated SKTextAnimator-related functions with SKText and SKLocalization

V0.9.3

- Added SKMeasurer to Physics&GO module.
- Added several icons.

**v0.8.x**

2022.4

- Added MathLibrary
- Added MonoCore & MonoBase to Common Module
- Added SKCell.cginc to Graphics Module
- Fixed SK asset path bugs

0.8.1

- Optimized Localization Module. Added subpages.
- Fixed font chart bugs.

**v0.7.x**

2022.3

- Added Event module (EventDispatcher, EventHandler, EventRef)
- Added Effect module (SKSpriteProcessing, SKImageProcessing, SKSpriteBlur)
- Added Sprite Colorer to Editor Module
- Removed DOTween from the package

V0.7.1

- Fixed editor namespace problems with SKGridLayer.

    - Fixed SKCore building errors.
    - Optimized SKUtils.StartProcedure method.

      V0.7.2
    - Added AnimationRandomizer to effect module

      V0.7.3
    - Added Light2D to effect module

      V0.7.4
    - Added several fonts to font module.
    - Reorganized package structure
    - Removed all third-party dependencies.

**v0.6.x**

2021.8

- Added FSM module. (FSM, FSMSystem, FSMTransition, FSMEvent)
- Added component icons.

  V0.6.1
    - Added ReplaceableMonoSingleton to common module.

      V0.6.2
    - Added MinHeap and MaxHeap to structure module.

      V0.6.3
    - Added ITreeNode, TreeNode, ITree, TreeStructure to structure module.

      V0.6.4
    - Added sorting utilities to SKUtils.

**v0.5.x**

2021.7

- Added grid module. (SKGrid, SKGridLayer, SKGridEditor, SKGridLayerEditor, SKGridOccupier, etc.)
  V0.5.1
  - Improved SKGrid performance.
  - Added SKSpriteTools to editor module.
    V0.5.2
  - Added SKTransformExt to editor module.
    V0.5.3
  - Added EditorCoroutineManager to editor module.
    V0.5.4
  - Added command pattern scripts. (ICommand, Command, CommandManager)

**v0.4.x**
2021.5

- Added native module. (NativeUtility, AndroidUtility, IphoneUtility)
  V0.4.1
  - Added SKCurve / SKCurveSampler to structure module.
    V0.4.2
  - Added SKSceneObjectResponder to Physics module.
    V0.4.3
  - Added SKObjectDragger, SKObjectRotater, and SKObject Scaler to Physics module.

# License