PathMagic getting started

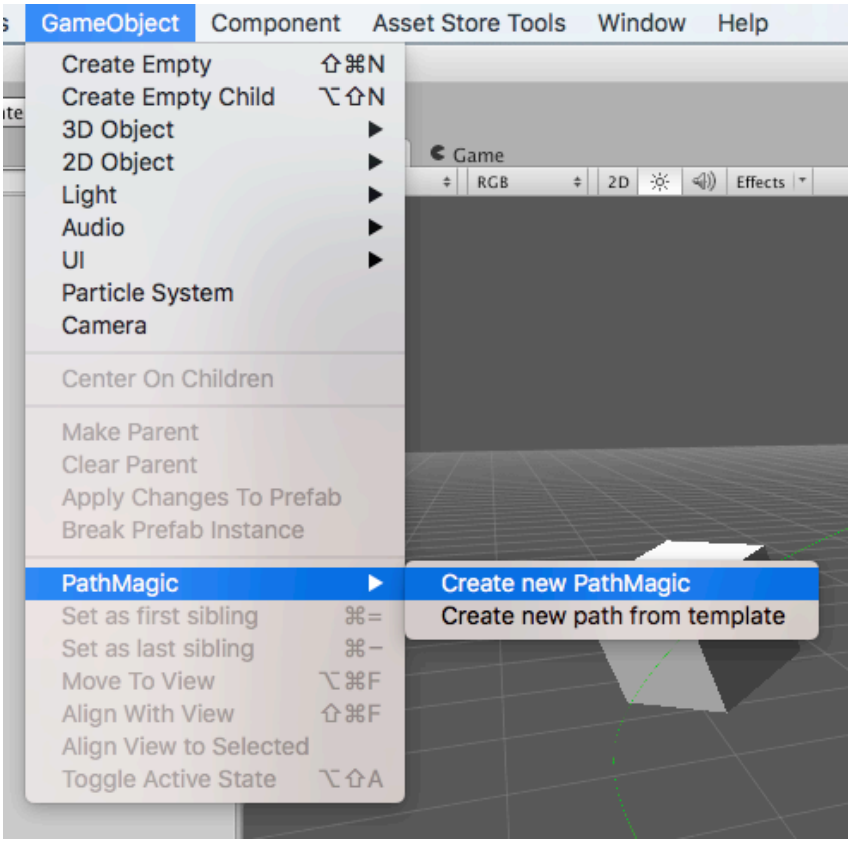
PathMagic is a UNITY editor extension that allows creating and managing fixed paths to move game objects smoothly in the scene. With PathMagic you can animate cameras, characters, buildings and other kinds of objects, in a fixed path defined by waypoints.

Each waypoint stores a velocity (at which the object will run), an input and output Bezier tangent, and a rotation criteria. Rotation can be specified manually or by following another target Transform.

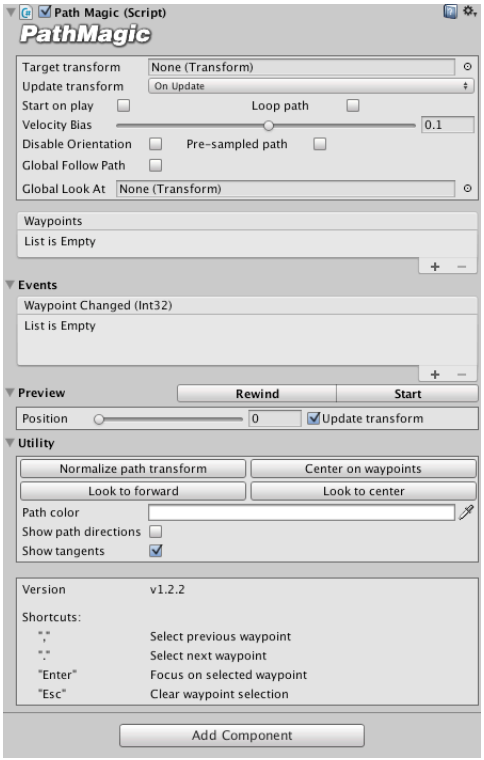
Let's start to create our first Path.

Create new PathMagic

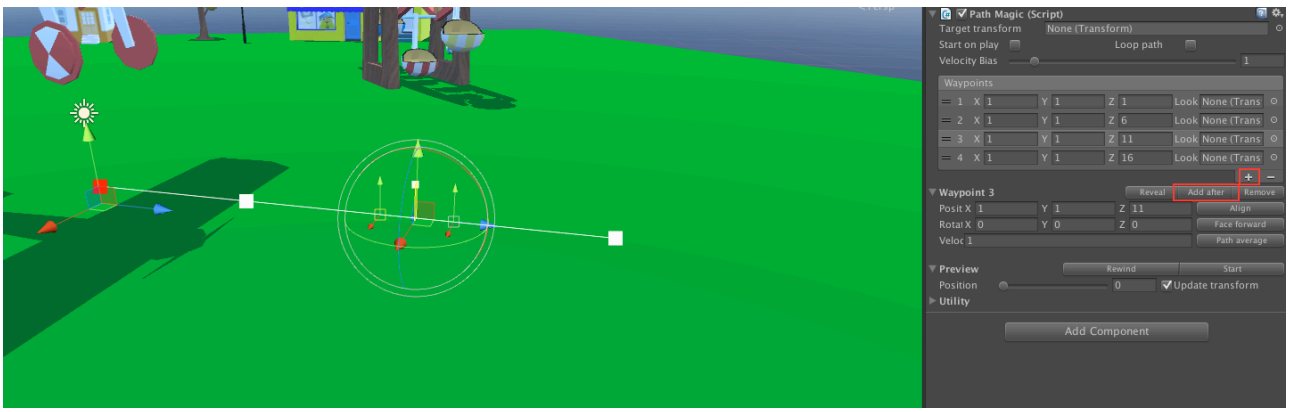
To create a PathMagic instance you can select the menu item "GameObject/PathMagic/Create new PathMagic". This will create a new Gameobject named "PathMagic Path" with the script "PathMagic.cs" attached. You can also attach the "PathMagic.cs" script to any other Gameobject in your scene, except the one that you want animate using the path itself.



Now you can work with the inspector to create and modify you path.



As the first step you have to add waypoints. By default waypoints are added to the Path transform beginning at v(5,5,0) and next at distance of 5 units from one to another, along the z axis. You can also add waypoints in the middle of the path by selecting an internal waypoint (clicking on a row in the waypoint list) and selecting the "+" (plus) button at the bottom of the list, or by clicking the "Add after" button.

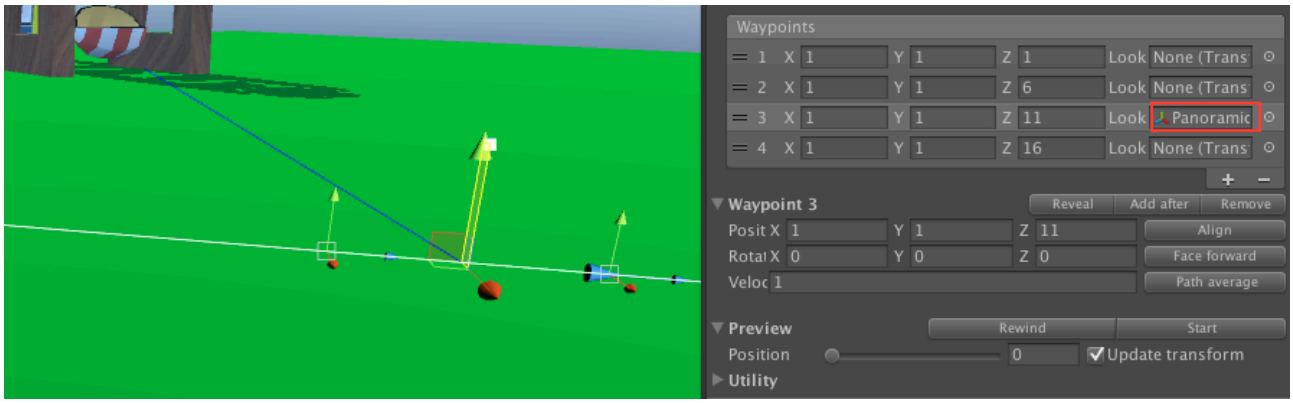


Waypoints shows these object in the scene and in inspector view

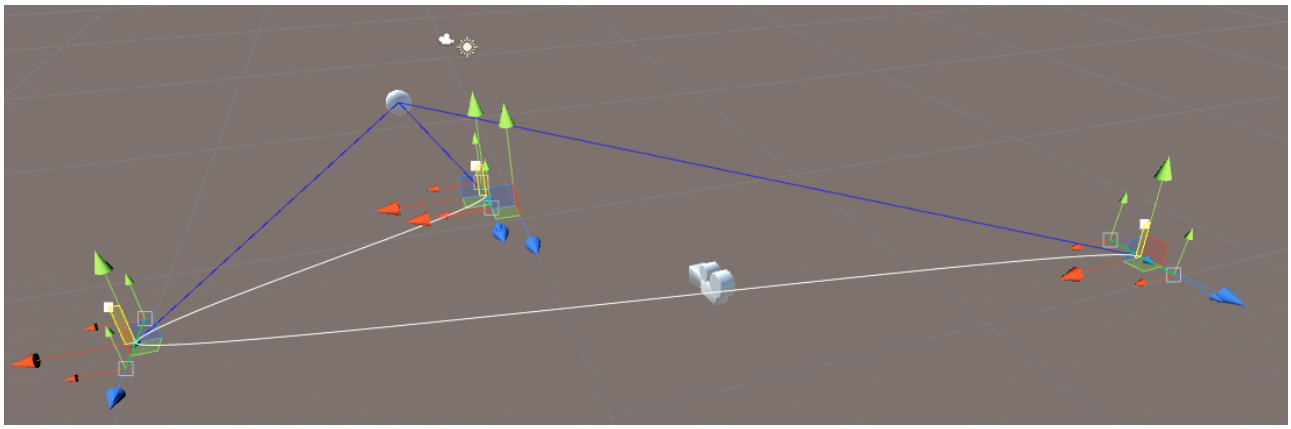
- A position handle to freely move the waypoint
- Two small symmetrical position handles to move Bezier control points
- A rotation angle to set orientation manually
- A velocity slider (yellow tower) to set the waypoint velocity

All values will be interpolated during the simulation. Orientation has specific cubic interpolation, so when specify orientation for a given waypoint (say waypoint “n”), keep in mind that waypoints n-1, n+1 and n+2 will have an influence in the rotation interpolation. So if you get a strange rotation evolution, try adjusting the other.

There are some special case for the rotation in a waypoint, you can specify a target Transform to follow in a given waypoint. In this particular case, the animated object will look at the Transform instead of assume the particular rotation. This allow to follow movable object.



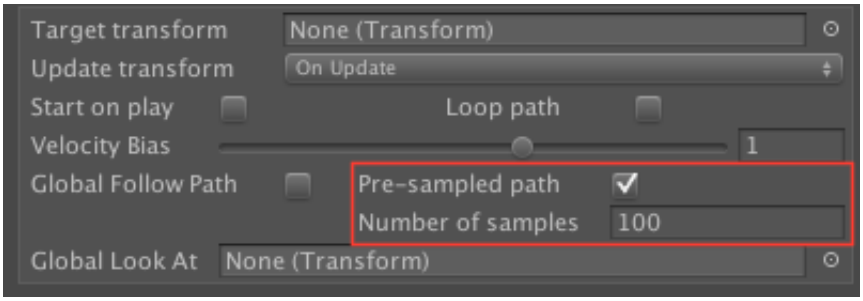
When you setup a target to follow, a blue line connecting the waypoint and the target will appear.



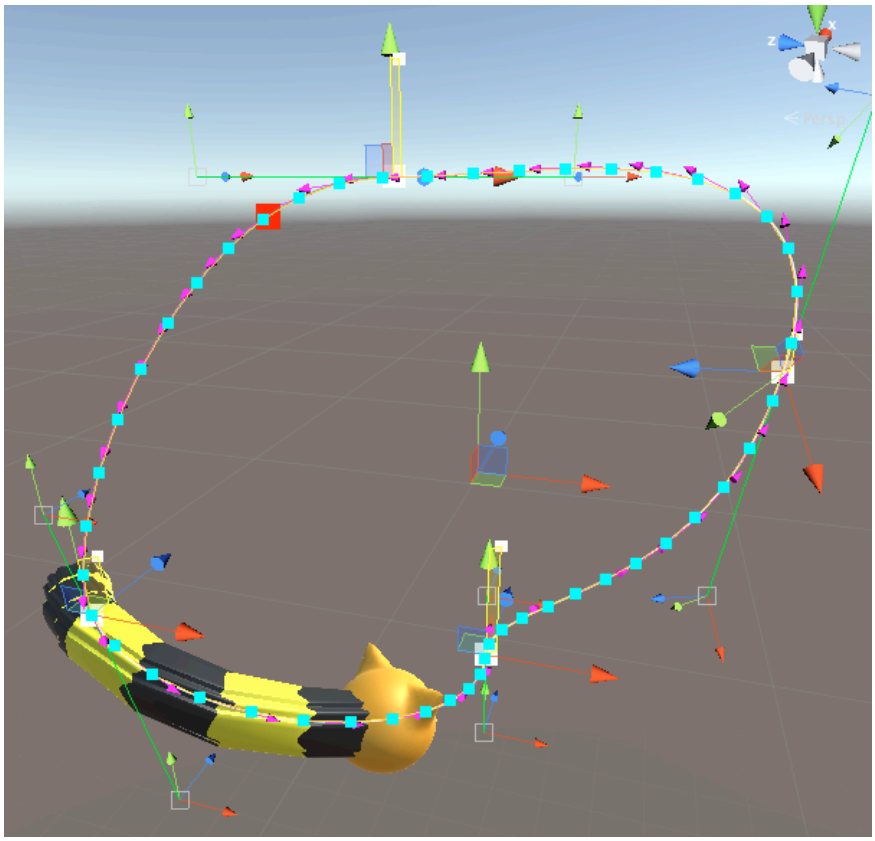
In the described case, the rotation assume a value that looks the specified target, but only when the path reaches the given waypoint, values before and after the waypoint are interpolated with previous and next rotation at waypoints. To simply follow the natural direction of the path you can set the “Global Follow Path” option.

Starting from v1.1.1 there is another useful field: Global Look At. If this field is set, the orientation of the transform is set to look at it, at each frame, instead of interpolating values between waypoints. If all waypoints rotation have to look to a single target, please use it, instead of “look” field at waypoint level as it is more precise.

Starting from v1.2 version there is an amazing feature: **Path Sampling**. Path sampling is the ability to pre-sample the path (and measure it) to give the possibility to run onto the path at constant velocity. To enable pre-sampling simply set the “Pre-sample Path” option. At this point you can modify the number of the samples (default: 100 samples).



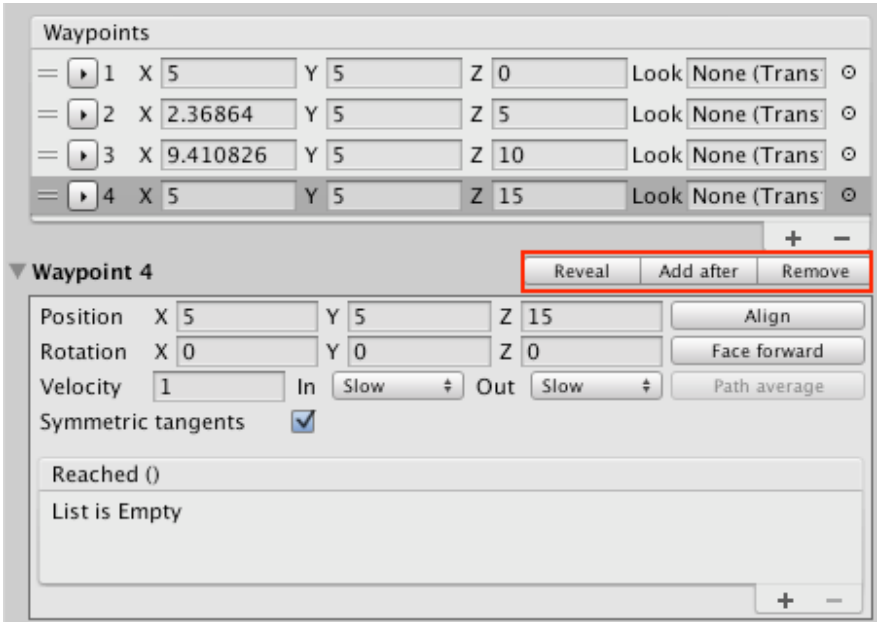
Enabling pre-sampling of path, you can see samples on the scene view.



Moving, rotating and scaling the whole path

Transform operation are translated by PathMagic in an interesting way. As you might guess, moving the path, will move all waypoints together. Rotating and scaling the whole path, will rotate and scale (enlarge) the path. This is a pretty useful behavior when need to adjust entire path to a particular situation.

Reveal, Add after, Remove, Align, Face forward and Path average

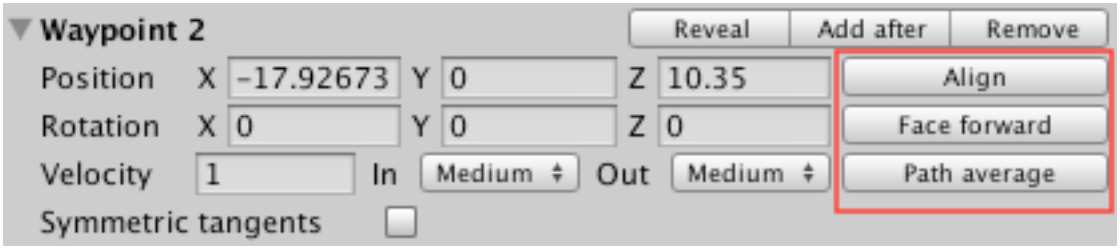


While a waypoint is selected, in the waypoints list, there are three simple action in the inspector:

1. **Reveal** – allow to select and pivot the specific waypoint in scene view
2. **Add after** – add another waypoint just after this one in the path sequence (equivalent to press the "+" button at the end of the waypoints list
3. **Remove** – simply remove the waypoint and select next in the sequence (equivalent to press the "-" button at the end of waypoints list

Other three buttons are:

1. **Align** – will align the position and tangents of the waypoint to best match the alignment with the previous and the next waypoint in the path sequence
2. **Face forward** – modify the waypoint rotation so that will points at the same direction as the path
3. **Path average** – modify the waypoint velocity by setting it at the average value between the previous and the next



In this section you can manually modify all waypoint properties, such as position, rotation (euler angles) and velocity. Starting from version 1.2, there is another option for each waypoint: "Symmetric tangents". You can enable/disable **Symmetric tangents** to create any type of Bezier path. If you unset it, you can create paths with hard edges.

Starting with version 1.2.1 there are some fields for adjusting the mode in which the velocity will varying from a waypoint to another. There are two fields near the velocity: **In** and **Out**.

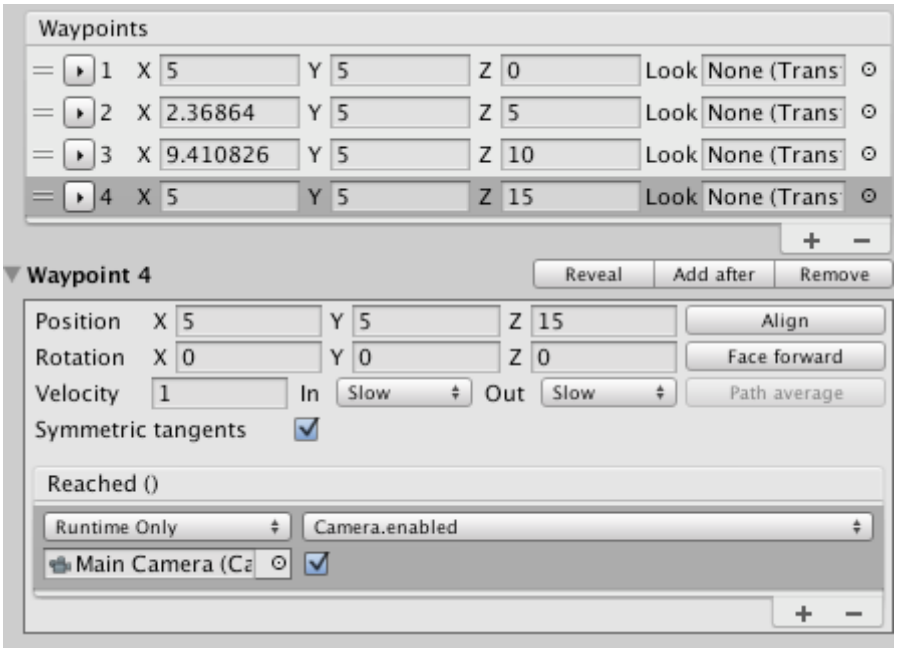
- **In** allows to specify how fast the velocity assume that the waypoint in input (from the previous waypoint). You have three possible values: **Slow**, **Medium**, **Fast**
- **Out** allows to specify how fast the velocity leaves that the waypoint in output (to the next waypoint) . You have three possible values: **Slow**, **Medium**, **Fast**

For two adjacent waypoints, specifying a *fast out* for the first and a *fast in* for the second, will interpolate the velocity from the first to the second in such a way as to arrive as soon as possible to the second velocity.

These parameters allows to specify something like ease in/out for "velocity" of the specified waypoint. Is a specification to interpolation from two adjacent velocities values.

Waypoint events

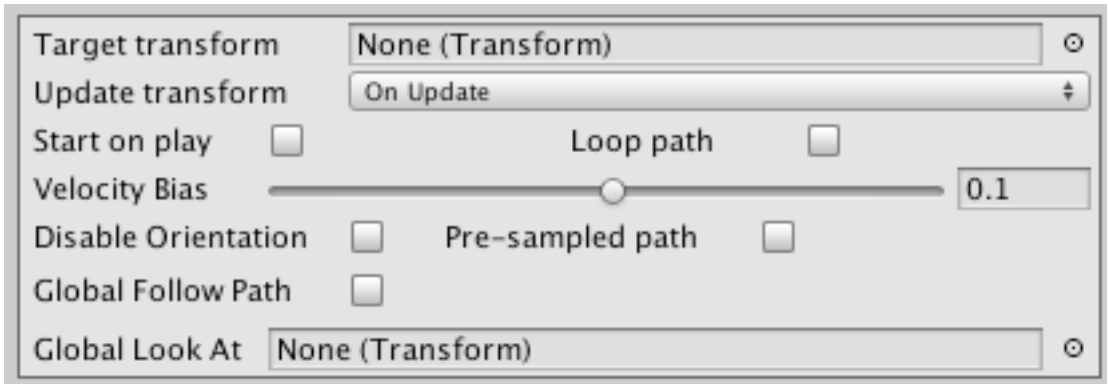
Each waypoint can fire events. Actually there is only one event fired: "Reached". The Reached event is fired when the animator reaches the waypoint. You can connect any method to this event to notify script that the animator has reached the waypoint.
This is useful for example to synchronize paths or synchronize a path with another game logic.



Global parameters

There are some global parameters:

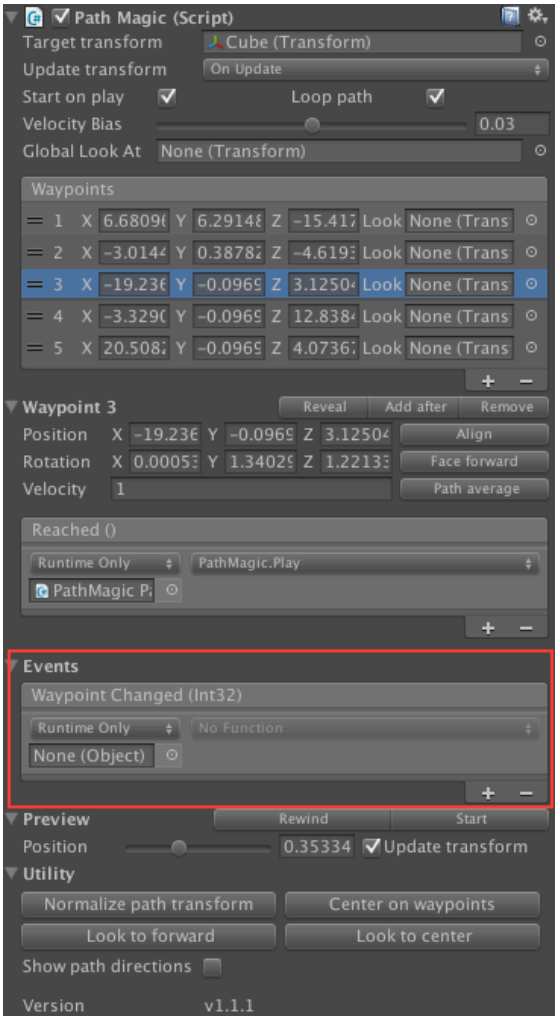
- **Target transform** – is the transform you want to animate on the path. Can be changed by scripting
- **Update transform** – Can assume "On Update" and "On Fixed Update". Set to Update to animate the target during the standard Update cycle, or "Fixed Update" if want to synchronize with physics.
- **Start on play** – if set, the animator starts automatically at the start of the scene. Otherwise you have to call `Play()` public function by scripting or modify **currentPos** public variable (clamped between 0f and 1f).
- **Loop path** – if set the path become infinite. When the end of the path is reached, the animator start again from the begin, interpolating values from the end and the start, otherwise, the animator will go automatically in `stop()`.
- **Velocity bias** – is a global regulation for the speed. Each waypoint continue to have its own velocity property related to this value. Starting from version 1.1.1 of the plugin, the velocity bias can assume negative values. A negative value causes the animation to play in reverse mode.
- **Disable Orientation** – Starting from version 1.2.2, you can prevent animators (PathMagic and PathMagicAnimator) from setting the orientation during animation. When this option is set, the target object is simply moved, and the orientation is leaved unchanged. This is useful if you want to manage the orientation directly, for example with mechanim.
- **Global Follow Path** – If set, the orientation will be set by following the natural path direction. This is a better way to specify a "face forward" at each waypoint, because the animator don't interpolate rotations between waypoints, but set the correct orientation at every frame
    - From version 1.2.1 of PathMagic, enabling Global Follow Path, will acetivate a new slider named **Follow Path Bias**. This slider allows you to adjust how smoothly the animator will follow the path. For animating cameras we suggest to set this value from mid-point to the end of value (0.1), it will results in more smooth camera view.
- **Global Look At** – If set, the orientation will be set by looking at this value. This is more a better way to looking the same target all the time, instead of specify the value at each waypoint "look" parameter, because the animator don't interpolate rotations between waypoints, but set the correct orientation at every frame



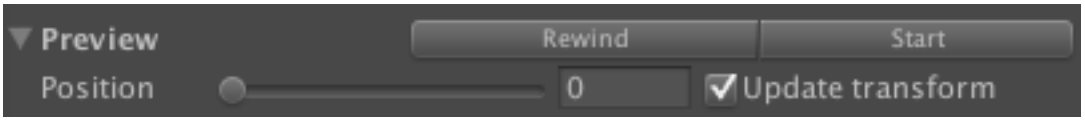
Global events

Global events are events fired when a global condition occurs. For now there is only one global event: "WaypointChanged (int)". This event is fired whenever the current (last passed)

waypoint has changed. When the event is fired, an integer parameter is passed too, representing the new current waypoint.



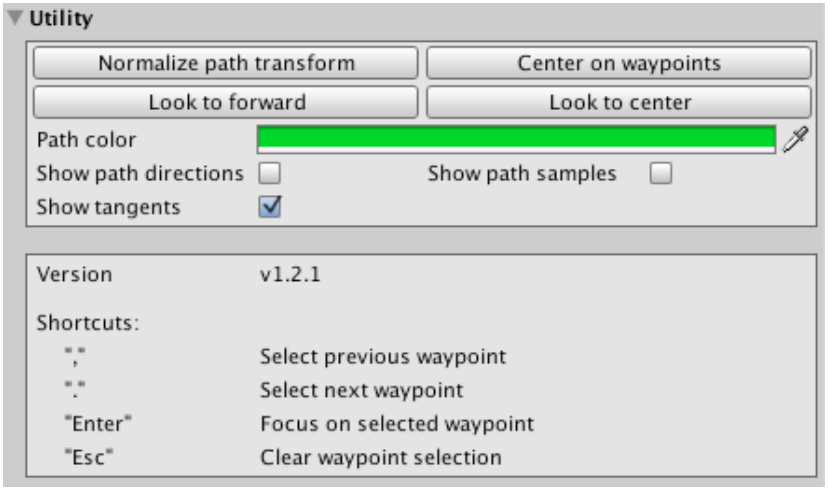
The preview section



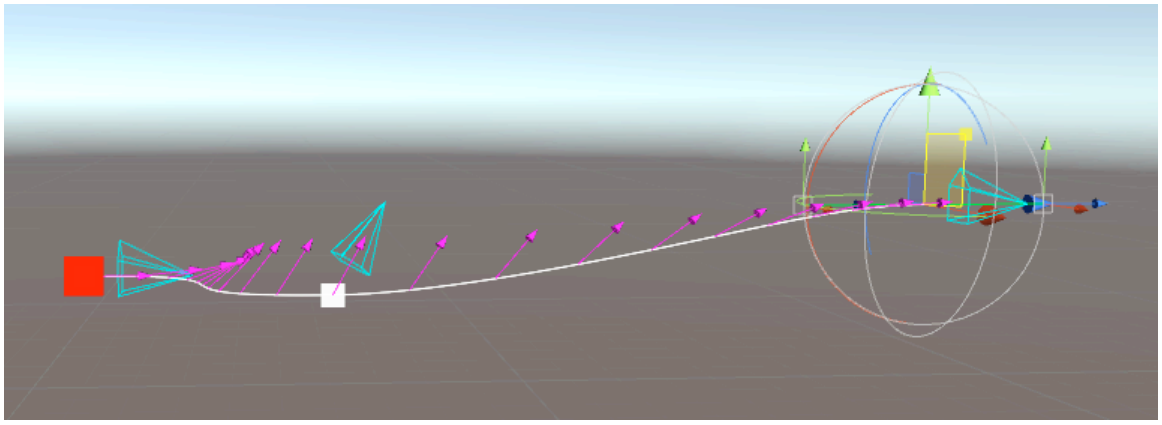
Preview section allows performing tests on the path. You can manually set the current position on the path, or start playing the animation in edit mode. During the animation you can however modify the current position to force the animation point. During the animation, you can freely modify / adjust the path, by modify and or adding new waypoints. This is pretty useful to speed up the path creation. Update transform option allows specifying whether or not to move the linked Transform.

Utility section

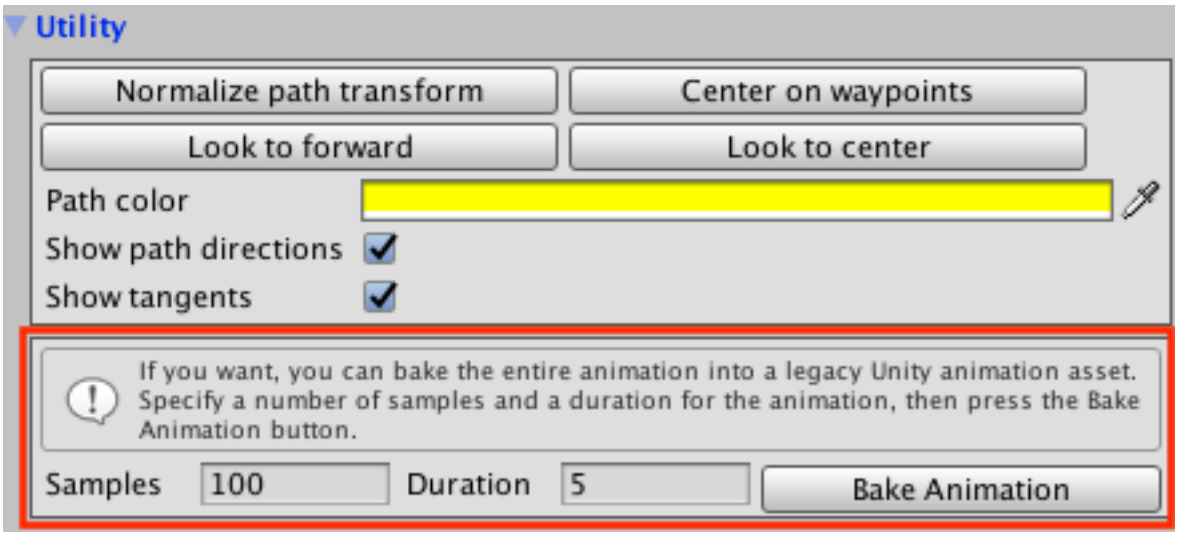
In the utility section there are some interesting global operation at path level.



- **Normalize path transform** – you can move, scale and rotate the gameobject's Transform to adjust the path, but in certain situation you may want to restore the Transform component without modifying waypoints. **Normalize path transform** does exactly that
- **Center on waypoints** – pretty useful when you want to move the transform position at the center of waypoints without modifying waypoints position. By centering on waypoints you can rotate the entire path around its center
- **Look forward** - will apply the Face forward waypoint operation to all waypoints of the path
- **Look to center** – modify the orientation of all waypoints so that all everyone will look towards the center of the path
- **Path color** – Allows to specify the color of the path (both in selected and not selected states). Coloring of the path may be useful to maintain a good visibility of path gizmos over any surface or complex scene (Thanks to Andrea)
- **Show path directions** – enable the scene editor to show orientations in a continuous way along the whole path (magenta arrows), as in the following picture
- **Show path samples** – Is set, only if the **pre-sample path** is on, show all computed samples (small cyan boxes)
- **Show tangents** – If set, the user can manipulate tangents in scene view. Users can unset this option to simplify graphical representation in case the path is pretty complex.

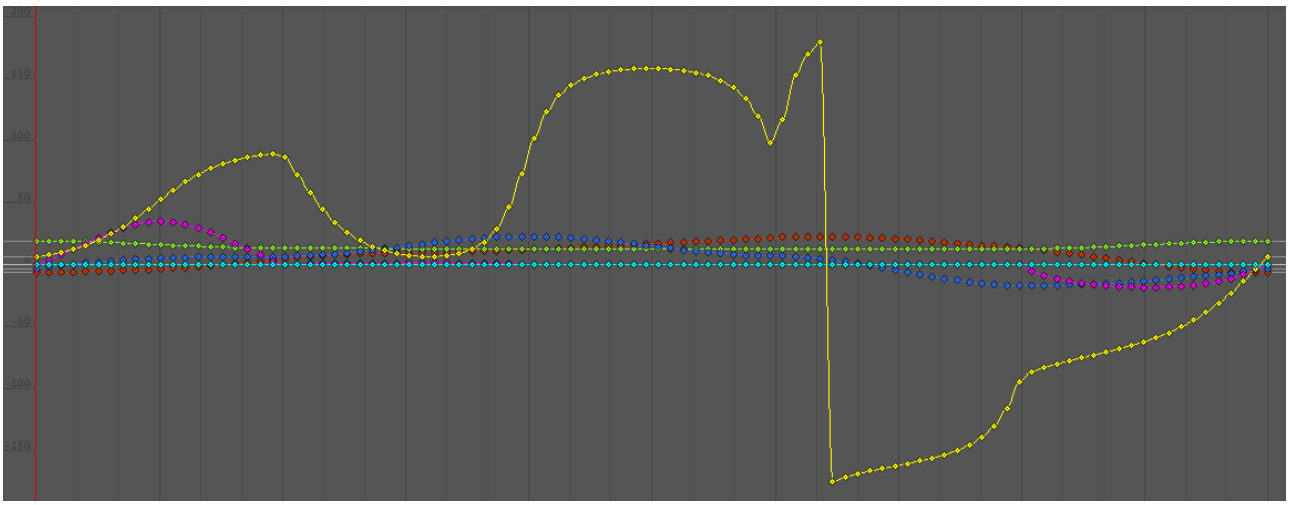


Starting from version 1.2.5, PathMagic support a Utility section for animation baking. In other words, you can bake the animation of the entire path into a legacy Unity animation asset. The utility section appears as follows.



You have to specify the number of samples (the number of key frames of the final animation), the duration (in seconds), and press the "Bake Animation" button. Unity will create the animation for the entire path (setting a loop animation if the path is a "loop path"), and attach it to the target object.

Once you have baked the animation, you can disable/destroy the PathMagic instance, because the animator will move the target object for you. In the following picture, you can see a key frame sequence generated by the bake animation process.



Keyboard support

Starting from version 1.2, the user can do some action with keyboard. For example you can "navigate" from a waypoint to the adjacent one by pressing "," and ".". Selecting a waypoint in scene view or in the inspector view, the user can focus it by pressing "Enter". By pressing "Esc" the selection is clear, no waypoint is selected, so by pressing "F" the natural focus on the PathMagic GameObject is executed.

Scripting

Scripting is very simple. Once the path was built, by scripting you can modify all public properties as well as call public functions of PathMagic and PathMagicAnimator instances. All classes in the package are placed under the "Jacovone" namespace.

Common methods of PathMagic and PathMagicAnimator:

- **`Play()`** –start the animation from the currentPos public variable
- **`Pause()`** – pause the animation without modifying currentPos
- **`Stop()`** – stop the animation and set currentPos to 0f
- **`Rewind()`** – equivalent to set currentPos to 0f without stopping the animation.
- **`GetCurrentWaypoint()`** – an integer value indicating the last passed waypoint index

Common properties:

Property	**In PathMagic**	**In PathMagicAnimator**
PathMagic	no	Yes
PathColor	Yes	No
Waypoints	Yes	No
Target	Yes	No
GlobalLookAt	Yes	Yes
GlobalFollowPath	Yes	Yes
GlobalFollowPathBias	Yes	No
UpdateMode	Yes	Yes
Loop	Yes	No
AutoStart	Yes	Yes
VelocityBias	Yes	Yes
CurrentPos	Yes	Yes
IsPlaying (RO)	Yes	Yes
WaypointChanged	Yes	Yes
PresampledPath (RO)	Yes	No
LastPassedWaypoint (RO)	Yes	No
PositionSamples (RO)	Yes	No
RotationSamples (RO)	Yes	No
VelocitySamples (RO)	Yes	No
WaypointSamples (RO)	Yes	No
SamplesDistances (RO)	Yes	No
TotalDistance (RO)	Yes	No

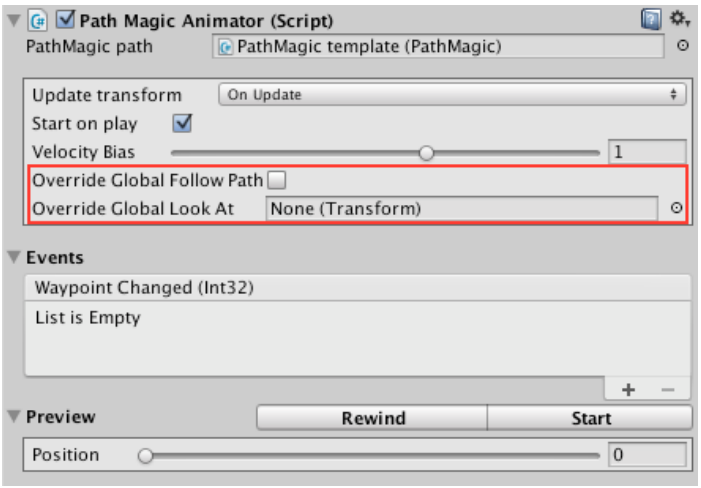
The script PathMagicAnimator

Starting from version 1.2 there is another script in the package: **PathMagicAnimator**. The script is a simple animator (as PathMagic is), but does not contain its own path, but use the path of another PathMagic instance.

This pattern allows the user to define a single PathMagic in the scene and attach an instance of PathMagicAnimator to each Transform that the user wants to animate in that path. The PathMagicAnimator references a PathMagic instance as its path definition.

So, for example, if you want animate some cubes on the same path (but a different velocities, positions, etc) you can create a classic PathMagic in the scene. After, create some 3D cubes in the scene, and on each cube attach the PathMagicAnimator script.

On each instance of these scripts, connect the original PathMagic and do some adjustments on the animator. Each animator is independent from others, but share the same path definition. This enable many possibilities, such as move bones of a skinned mesh renderer along the same path, to obtain impressive effects.



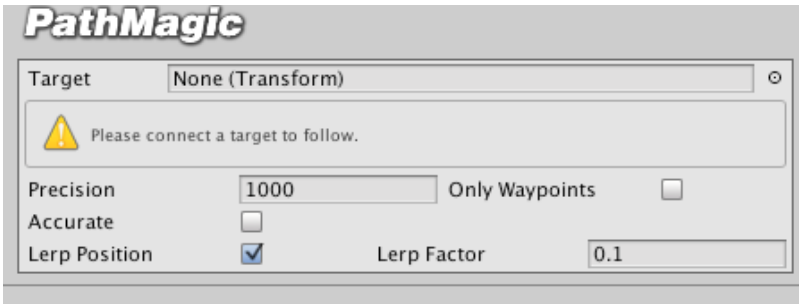
Each animator can manage its own events, as basic PathMagic do.
Starting from version 1.2.1 of PathMagic, PathMagicAnimator allows to set

- **Override Global Follow Path**, if you specify this option for this animator, the connected Transform will globally follow the path even the original path is not set up to do. This allows to have objects on the same path in which someone will follow the path while someone other does not.
    - For implementation related reasons, this option is available only for non pre-sampled original paths.

- **Override Global Follow Target**, if you specify a target for this animator, the connected Transform will globally follow the specified target even the original path is not set up to do. This allows to have objects on the same path in which someone will follow a target while someone other does not.

The script PathMagicFollower

Starting from version 1.2.2, the package contains the new script PathMagicFollower. The script can be attached to a GameObject with PathMagic or PathMagicAnimator component. Take a look at the inspector view of the script.



The feature allows the target connected to the path, to be at the minimum possible distance from another Transform in the scene. Consider a camera locked on a Path (for example if the camera is the target of a PathMagicInstance, or the camera have a PathMagicAnimator instance attached). You want the camera stay on its own path, but follow a specific target. This script allows the camera to "navigate" the path following the specified target.
To setup this scenario,

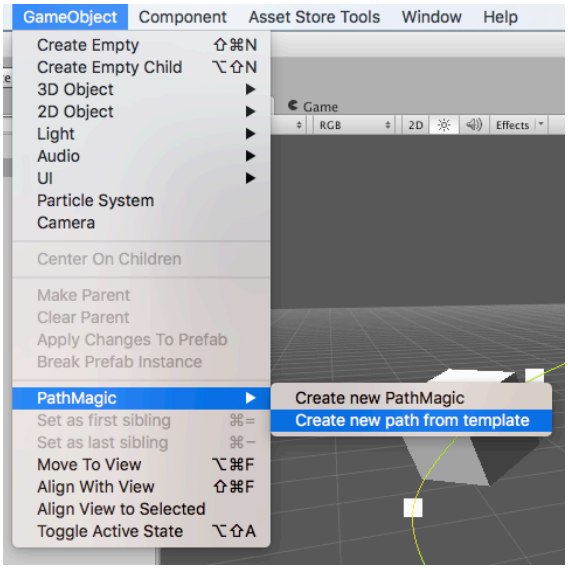
1. Create a new PathMagic instance
2. Set the main camera as target (moved) object
3. Attach PathMagicFollower to the PathMagic instance
4. Set a simple transform as target object on the PathMagicFollower inspector
5. Try to move the target Transform (also in edit mode)

There are some properties of PathMagicFollower

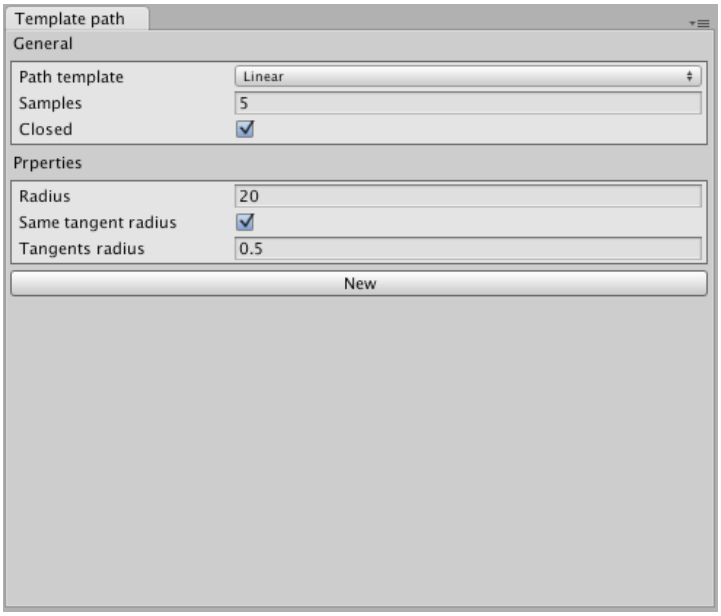
- Precision – To follow the target the script compute the minimal distance from it, along entire path, to do this, the path have to be subdivided. This parameter is the number of subdivision steps. Greater values of this parameter gives more accurate results but have greater computation costs
- Only Waypoints – If set, the PathMagicFollower component will move the target only from a waypoint to another (smoothly or not, depending on next property)
- Lerp Position – If set the movement is "lerped". The false value of this property makes sense only if "Waypoints only" is true. In fact, in this case, the object is instantly moved from a waypoint to another in a frame. This is useful if you use a camera to follow a target.
- Lerp Factor – Defines the factor of the previous Lerp operation for move the target on the path.
- Accurate – Gives additional precision to the minimum distance point from the path and the target object. Set it to true if you experience some small jumps on object movement along the path while follow the target.

Path Templates

Starting with version 1.2.1 of PathMagic, the user can start to build a path with a new tool: Path Templates:



The tool is an editor floating window that the user can dock wherever it wants.

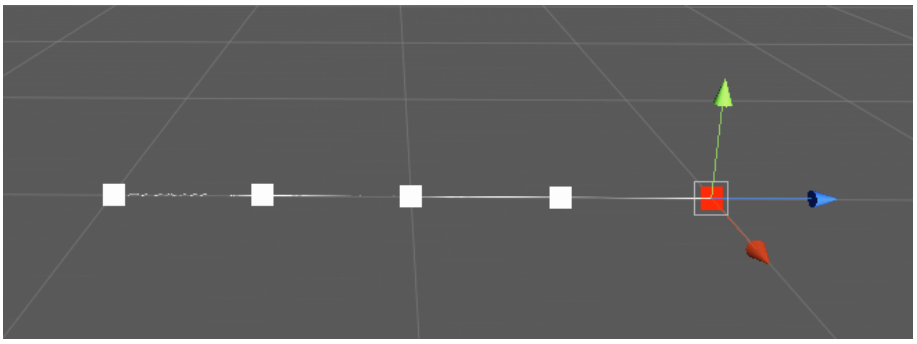


There are two type of path templates: **Linear** and **Circular**. Regardless of which option you choose, there are two generic properties:

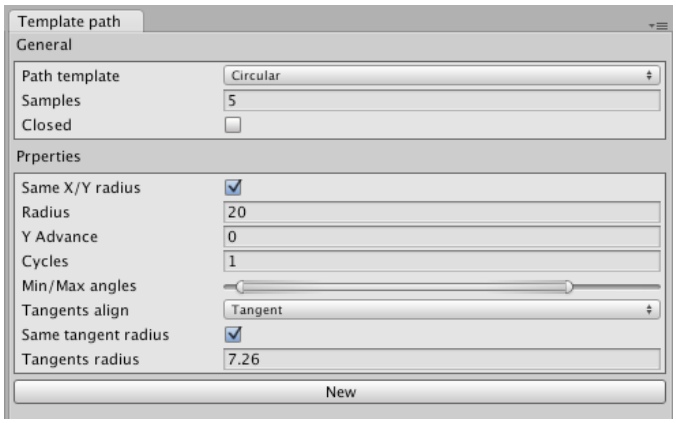
- **Samples** – the number of waypoints being generated
- **Closed** – if set, the generated path will be closed

In the previous picture, the Linear mode is selected, in this mode available properties are those in the picture.

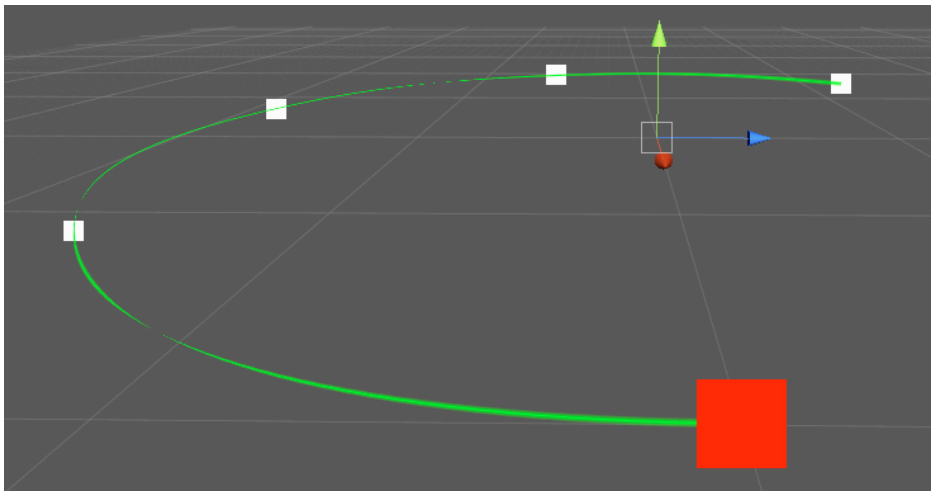
- **Radius** – the length of the path in world units
- **Same Tangent Radius** – have in and out tangents the same radius (length)?
- **Tangent Radius** – the tangent radius (length). If previous option is not set, this property splits itself in two: **In Tangent Radius** and **Out Tangent Radius**.



In Circular mode there are other properties.



- **Same X/Y Radius** – allows to have different X/Y radius. This can be useful for generating ovals
- **Radius** – is the radius of the circular path. If the previous option is not set, it splits itself in two: **Radius X** and **Radius Y**.
- **Y Advance** – is an optional advance amount in the up direction. It allows to generate spirals in conjunction whit following (Cycles)
- **Cycles** – allows to specify how many times the path turns around.
- **Min/Max angles** – useful to specify a simple arc. Values are in the range [0..360]
- **Tangents align** – there are two options
    - **In Path** – tangents are placed exactly in the direction of previous / next waypoint. It generates polygons
    - **Tangent** – tangents are placed to create a circular path.
- **Same tangent radius** – give the option to have a different radius for in/out tangents
- **Tangent radius** – the radius (length) of generated tangents. If the previous option is not set, this option splits itself in two: **In tangent radius** and **Out tangent radius**.



The **New** button, detaches the current generated path from the editor window and create a new template path (linear) in the scene view. You can also close the editor window to leave the generated path in the scene view.

Playing with parameters allows you to create several regular path, try it!

Support

To request support, write to: **marco.jacovone@gmail.com** or **marco@jacovone.com**.

Thanks…

- To **Aaron** for suggestions and active participation
- Thanks to **Leon** for contributing significantly to make this product better
- Thanks to **Mads** for your contributing in beta testing
- Thanks to **Andrea** for its useful suggestions and active participation
- Thanks to Mats Nielsen for software and documentation bug solution