CAROUSELPRO DOCUMENTATION

Introduction

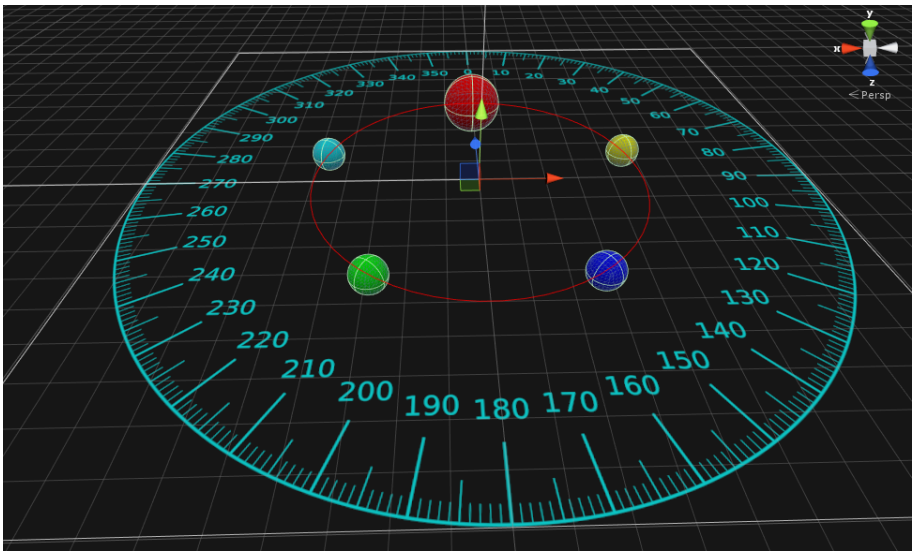
First, thank you for purchasing CarouselPro. Before starting to use this package, you need to know that it **requires at least the free version of DoTween** for all the animations to work. If DoTween hasn't been imported yet you'll have a compilation error. DoTween is full of features and is a great time saver so we encourage you to use it for your projects. Thanks to Daniele Giardini (aka Demigiant) for this great package!

CarouselPro lets you insert objects into slots which are evenly distributed along a circle (clockwise or counterclockwise) of configurable radius. A slot is either selected or not and selection can be changed through script. Selection change is animated by rotating the carousel by default but this can be de-activated. A different scale factor, position offset or rotation speed can be applied to a slot depending on its selection state and the transition between these states can be animated. The carousel's visibility (game object active state) can be changed and this transition can also be animated. Callbacks can be provided when changing the selection or changing the visibility of the carousel and are invoked when animation completes. Alternatively, other scripts that don't trigger change of selection/visibility themselves can register for those events.

The main component of the Carousel is a list of objects. Some functions of this list have been made available: Add, AddRange, Remove, RemoveAt, etc. It should work as expected with some special semantics in certain cases. A notable example is that you can't have duplicates in the list as an object cannot be inserted in different slots at the same time. Check the reference below for more information.

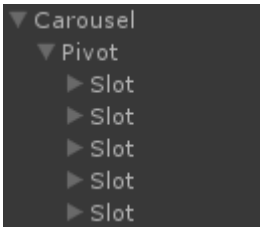
Quick tour

This package has three examples that demonstrate what can be done with the current version of CarouselPro. You can find them in the "Sidema/CarouselPro/Examples" folder.



Scene **Example1** shows a simple carousel with a protractor attached to it to show the way each object is spread along a circle at regular angle intervals.

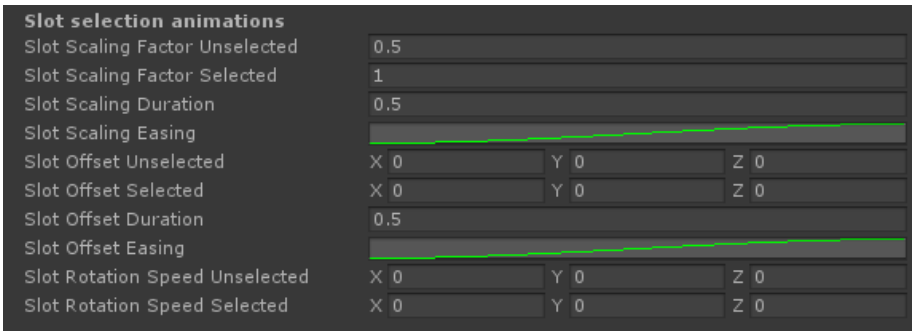
Each object is attached to a helper object called "**Slot"** that represents the slot into which the object is inserted. An object stays untouched while inserted into the carousel (except for its position of course) and can be added/removed at any moment. Animations are applied to objects through their corresponding slot objects whereas carousel rotation uses another helper object called "**Pivot"** to which all slots are attached.



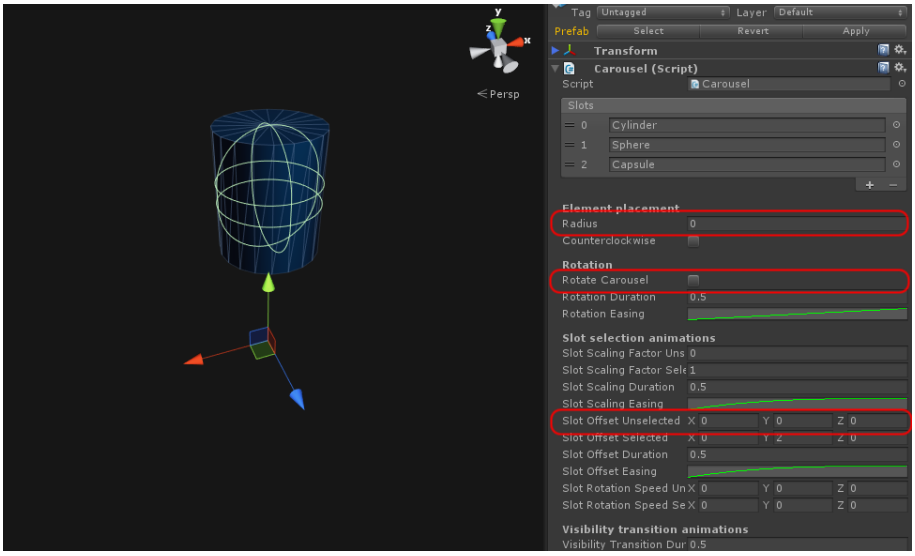
Pressing the left or right arrow keys will change the selected slot. This is made possible by the example script ***SingleCarouselController*** which demonstrates the usage of the carousel's functions ***Next*** and ***Previous***. If you expand the carousel hierarchy, you will notice the helper object called "Pivot" rotating. It makes slots and any child object you added to it to rotate as well. Here our protractor is rotating because it was added as a child of this Pivot object. It is perfectly legit but, when manipulating the hierarchy, simply make sure not to mess with slot objects.

If rotation is enabled (via *Rotate Carousel* property), the currently selected slot is always aligned with the z-axis of carousel root object (see Example2 if you don't want the carousel to rotate when changing selection). In this example the z-axis of the carousel is pointing away from the camera so selected slot is in the upper part of the view.

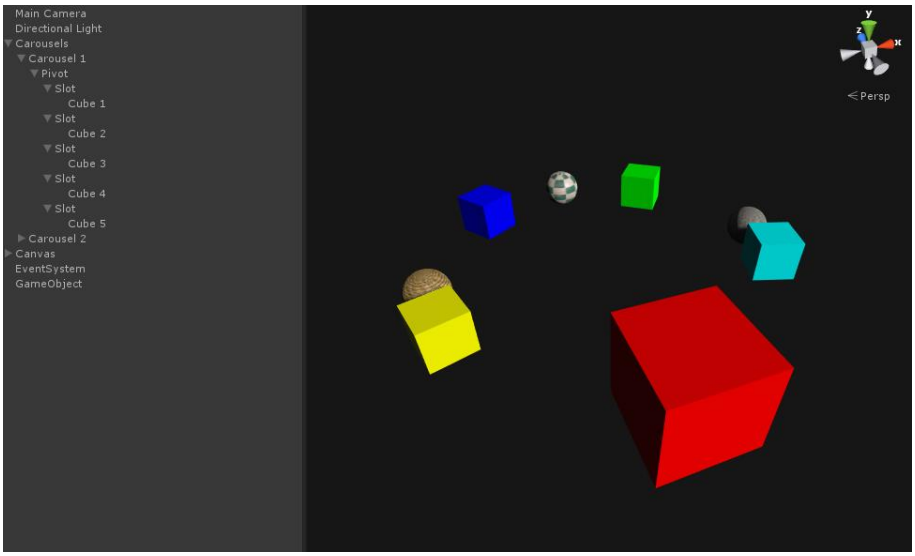
You can see that selected slot's object is "highlighted" by being made bigger than unselected ones, this can be personalized in the carousel inspector with the properties under the **Slot selection animations** header. There you can find multiple properties with the "selected" and "unselected" term in their name, you can change any of them to see the changes in real time. All the selected/unselected properties have duration and easing curve properties associated which allow customization of the animation for selection state transition. By default, the easing curve is linear as if no easing was applied.



In the scene you will also find an object called ***CarouselTest***. A script with the same name is attached to this object and showcases the different list-manipulating functions of the Carousel class.

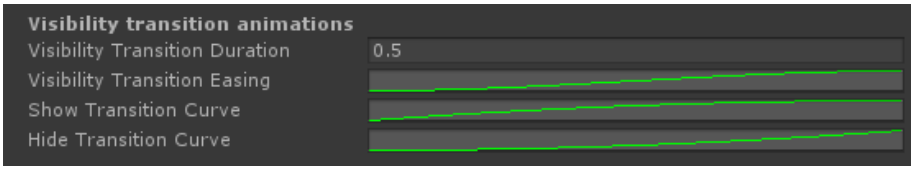


Scene **Example2** demonstrates that the default behavior of the carousel, which is to rotate when changing selection, can be deactivated by unchecking the **Rotate Carousel** property. This allows for different effects. Here we also set the "Slot scale factor Unselected" to 0 so that unselected slots "disappear" completely. We also set the offset of selected slot to be 2, so that selected object seems to appear by popping up. Easing curves for these two transitions are set to accentuate that popping effect.



Scene **Example3** shows how one can use the visibility transitions to combine multiple carousels and switch between them. Here two carousels have been added as children of another game object on which the ***MultiCarouselsController*** script is present. The script is automatically finding any child carousel and lets you switch between those. The script also uses the callback mechanism to check for selection change and visibility state.

Visibility transitions types are NONE, UP or DOWN. Transitions animations are only applied when using the UP and DOWN types. The animation is applied to the carousel through its pivot (like for rotation) and can be customized in the inspector with properties under the **Visibility transition animations** header.



Easing property provides the easing applied to the main transition animation which is to move the carousel pivot up from -radius to 0 for UP animation or down from radius to 0 for DOWN animation.

Curve properties are used by the transition sub-animation and can be different whether we show or hide the carousel. For UP and DOWN those curves provide easing to a scaling animation. If you set linear curves the carousel will scale up (down) regularly while showing (hiding).

Carousel script reference

Properties

counterclockwise	True if objects are distributed in a counterclockwise fashion.
radius	The radius of the circle along which slots are distributed.
rotateCarousel	True if the carousel rotates when selection change ; false otherwise.
rotationDuration	The duration of the carousel's rotation animation.
rotationEasing	The easing curve for rotation animation of the carousel.
slotScalingFactorUnselected	The scaling factor applied to unselected slots.
slotScalingFactorSelected	The scaling factor applied to selected slot.
slotScalingDuration	The duration of the slot scaling animation.
slotScalingEasing	The easing curve used for slot scaling animation.
slotOffsetUnselected	The offset applied to unselected slots.
slotOffsetSelected	The offset applied to selected slot.
slotOffsetDuration	The duration of the slot offset animation.
slotOffsetEasing	The curve used to ease the slot offset animation.
currentRotation	Current rotation of the carousel.
angleIncrement	The angle increment between two objects.
count	The number of slots.
selectedObject	The object inserted in the selected slot. Can be null if the slot is empty.
selectedSlotIndex	The selected slot index. -1 if no slots are present in the carousel.
slotObjectPoolSize	The slot object pool size.

Events

onSelectionChanged	Invoked when selection changed and animation is complete.
onShow	Invoked when the carousel is shown and animation is complete.
onHide	Invoked when the carousel is hidden and animation is complete.

Methods

Contains

void **Contains**(GameObject **gameObject**)

gameObject	The object to locate.

Determines if a given object is in the carousel.

IndexOf

void **IndexOf**(GameObject **gameObject**)

void **IndexOf**(GameObject **gameObject,** int **index**)

void **IndexOf**(GameObject **gameObject,** int **index,** int **count**)

gameObject	The object to locate.
index	The starting index of the search.
count	The number of slots to search into.

Returns the slot index of the given object, or -1 if it cannot be located.

Search can be made on the whole range of carousel slots or on a specified range.

Insert

void **Insert**(int **index**, GameObject **gameObject**)

index	The index where the object should be inserted.
gameObject	The object to insert.

Inserts an object in a new slot at given index. If the object is already present in the carousel, it simply moves to the new slot leaving previous slot empty.

InsertRange

void **Insert**(int **index**, IEnumerator<GameObject> **gameObjects**)

index	The index where to insert the objects.
gameObjects	The objects to insert to the carousel.

Inserts a collection of objects in the carousel in new slots from given index. Inserting objects already present in the carousel will simply move them to the new slots leaving previous slots empty.

Add

int **Add**(GameObject **gameObject**)

gameObject	The object to add.

Adds an object to the carousel in a new slot. Adding null will create an empty slot.

Adding an object already present in the carousel will simply move it to the new slot leaving previous slot empty.

Returns the index of the slot into which the object has been added.

AddRange

void **AddRange**(IEnumerable<GameObject> **gameObjects**)

gameObjects	The objects to add to the carousel.

Adds a collection of objects to the carousel in new slots. Adding objects already present in the carousel will simply move them to the new slots leaving previous slots empty.

Remove

void **Remove**(GameObject **gameObject**)

gameObject	The object to remove.

Removes the given object from the carousel if present and delete its slot. The removed object is un-parented from the carousel.

RemoveAt

void **RemoveAt**(int **index**)

index	The slot index.

Removes the object from the slot at given index (if any) and delete its slot. The removed object (if any) is un-parented from the carousel.

RemoveAllEmpty

void **ClearEmpty**()

Removes all empty slots from the carousel.

Clear

void **Clear**()

Removes all the objects from carousel. Objects are un-parented from the carousel.

Get

GameObject **Get**(int **index**)

index	The slot index.

Returns the object in the slot at given index or null if the slot is empty.

Set

void **Set**(int **index,** GameObject **gameObject**)

index	The slot index.
gameObject	The object to set.

Sets the object in the slot at given index to the given object.

CopyTo

void **CopyTo**(GameObject[] **array**)

void **CopyTo**(GameObject[] **array**, int **index**)

array	The array to copy objects to.
index	The index from where to start copying in the array.

Copies the objects in the carousel to an array. You can specify a beginning index in the target array.

GetEnumerator

IEnumerator **GetEnumerator**()

Returns an enumerator that iterates through the carousel objects.

Previous

void **Previous**(Action **onSelectionChanged** = null)

onSelectionChanged	Callback invoked when selection has changed and animation is complete.

Selects the previous slot in the carousel (cycle to the last slot if current selection is the first slot).

Next

void **Next**(Action **onSelectionChanged** = null)

onSelectionChanged	Callback invoked when selection has changed and animation is complete.

Selects the next slot in the carousel (cycle to the first slot if current selection is the last slot).

This has no effect when there are no objects in the carousel.

SetSelection

void **SetSelection**(int **index**, Action **onSelectionChanged** = null)

index	The slot index.
onSelectionChanged	Callback invoked when selection has changed and animation is complete.

Selects the slot at given index.

Show

void **Show**(VisibilityTransitionType **transition** = VisibilityTransitionType.NONE, Action **onComplete** = null)

transition	The type of visibility transition to use.
onComplete	Callback invoked when animation is complete.

Shows the carousel with an optional transition animation and an optional callback invoked when animation is complete.

Hide

void **Hide**(VisibilityTransitionType **transition** = VisibilityTransitionType.NONE, Action **onComplete** = null)

transition	The type of visibility transition to use.
onComplete	Callback invoked when animation is complete.

Hides the carousel with an optional transition animation and an optional callback invoked animation is complete.

ClearSlotObjectPool

void **ClearSlotObjectPool**()

Clears the slot object pool used to store unused slot objects.