Object Placement

Table of Contents

Introduction

Here you can read and quickly find code documentation of the asset package.
For guide you can check out the set up guide and how it works section here:
https://travljen.gitbook.io/object-placement/getting-started/add-object-placement-to-scene

Script Reference

Scripts

ObjectPlacement.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Enumerations

GroundAlignmentType

```
enum GroundAlignmentType{
     None,
     ValidateOnly,
     AlignToGround,
     LowestPoint}
```

Defines the alignment type of placing objects.

None: Object will not be aligned to ground, nor will they be validated.

ValidateOnly: Object will not be aligned to ground. It will only be validated based on rotation threshold.

AlignToGround: Object will align to ground with its edges touching the mesh or terrain below it, rotating X and Z axis to achieve it.

LowestPoint: Object will find which of its edges is the lowest on ground and position itself on it, without rotation.

Classes

ObjectPlacement

```
public class ObjectPlacement
: MonoBehaviour
```

Variables

_instance

```
private static ObjectPlacement _instance
```

playerCamera

```
[Tooltip("Specifies the main camera which renders world to the
player.")]
[SerializeField]
private Camera playerCamera
```

placementObjectPrefab

```
[Tooltip("Specifies the prefab used for placement. This may also be
set during runtime, when the placement is not active.")]
[SerializeField]
private GameObject placementObjectPrefab
```

groundLayer

```
[Tooltip("Specifies the ground layer mask for ground detection. " +
"Use the layer on which the objects should be placed,
commonly ’Ground’ is used for terrains or ground planes.")]
[SerializeField]
private LayerMask groundLayer
```

maxDetectionDistance

```
[Tooltip("Specifies the maximal range allowed for ground detection
from camera/cursor position.")]
[SerializeField, Range(10, 5_000)]
private float maxDetectionDistance
```

obstacleLayer

```
[Tooltip("Specifies the obstacles layer mask used for detecting for
any " +
"objects that might be in the way, like environment or
already placed objects.")]
[SerializeField]
private LayerMask obstacleLayer
```

destructablesLayer

```
[Tooltip("Specifies the layer of destructables used for detecting
objects in placement bounds that may be destroyed when object is
placed.")]
[SerializeField]
private LayerMask destructablesLayer
```

destroyDestructableOnPlacement

```
[Tooltip("Specifies if objects detected as destructable obstacles are
destroyed on placement." +
"If this is set to false, you need to manage
destruction with OnObjectDestroy event.")]
[SerializeField]
private bool destroyDestructableOnPlacement
```

ignoreWhenOverUI

```
[Tooltip("Specifies if the placement should be ignored when cursor is
hovering over UI components.\n\n" +
"Example: If this is disabled, clicking a button on UI
when placement is active will still trigger placement." +
"When enabled, such placement action will be ignored.")]
[SerializeField]
private bool ignoreWhenOverUI
```

placementArea

```
[SerializeField]
[Tooltip(
"Specifies the area of valid placement. This can restrict
placement of objects outside of certain area.")]
private PlacementAreaInfo placementArea
```

groundAlignmentType

```
[Tooltip("Specifies if the object should visually align to the ground
instead of the placement center. See 'GroundAlignmentType' for more
information.")]
[SerializeField]
private GroundAlignmentType groundAlignmentType
```

groundAlignment

```
[SerializeField]
private GroundAlignment groundAlignment
```

groundAngleThreshold

```
[Tooltip("Specifies the angle range in which the object placement is
valid, " +
"once angles go out of this bounds, placement is no
longer valid. " +
"This ensures that the ground below it is flat enough for
placement of objects.")]
[IntRange(0, 90)]
[SerializeField]
private IntRange groundAngleThreshold
```

rotationSpeed

```
[Tooltip("Specifies rotation speed. If holdToRotate is enabled, then
this value will be multiplied with delta time on each update.")]
[SerializeField]
private float rotationSpeed
```

preserveRotation

```
[Tooltip("Specifies if the rotation between placements is preserved.
" +
"If a placement is canceled or finished, should the next
placement keep the Y rotation or reset to 0.")]
[SerializeField]
private bool preserveRotation
```

usePositionSnapping

```
[Tooltip("Specifies if the position snapping feature is enabled.")]
[SerializeField]
private bool usePositionSnapping
```

gridSnapper

```
[SerializeField]
private GridSnapper gridSnapper
```

applyPlacementMaterials

```
[Tooltip("Specifies if valid/invalid materials are applied to the
placing object.")]
[SerializeField]
private bool applyPlacementMaterials
```

validPlacementMaterials

```
[Tooltip("Specifies a list of materials applied to placing object
when placement is valid.")]
[SerializeField]
private Material[] validPlacementMaterials
```

invalidPlacementMaterials

```
[Tooltip("Specifies a list of materials applied to placing object
when placement is invalid.")]
[SerializeField]
private Material[] invalidPlacementMaterials
```

allowPlacementDuringAnimations

```
[Tooltip("Specifies if placing an object is allowed during active
movement or rotation animations. " +
"When this is disabled ‘animatingAngleThreshold‘ and
‘animatingDistanceThreshold‘ will be used.")]
[SerializeField]
private bool allowPlacementDuringAnimations
```

animatingAngleThreshold

```
[Tooltip("Specifies what the max angle difference can be between
target rotation " +
"and current rotation, when placement during animation is
not allowed.")]
[SerializeField, Range(0, 360)]
private float animatingAngleThreshold
```

animatingDistanceThreshold

```
[Tooltip("Specifies the max distance allowed between target position
and current placing object, " +
"when placement during animation is not allowed")]
[SerializeField, Range(0, 100)]
private float animatingDistanceThreshold
```

animatePositionChange

```
[Tooltip("Specifies if the position changes are animated. This means
that if enabled, " +
"placing object will follow cursor with position change
speed.")]
[SerializeField]
private bool animatePositionChange
```

positionChangeSpeed

```
[Tooltip("Specifies the speed of position change if the change is
animated.")]
[SerializeField]
private float positionChangeSpeed
```

animateRotationChange

```
[Tooltip("Specifies if the rotation changes are animated. This means
that if enabled, " +
"placing object will rotate with the rotation change
speed.")]
[SerializeField]
private bool animateRotationChange
```

rotationChangeSpeed

```
[Tooltip("Specifies the speed of rotation change if the change is
animated.")]
[SerializeField]
private float rotationChangeSpeed
```

debugLogsEnabled

```
[Tooltip("Specifies if the console logging is enabled.")]
public bool debugLogsEnabled
```

drawGizmosEnabled

```
[Tooltip("Specifies if the gizmos for placement are enabled.")]
public bool drawGizmosEnabled
```

gizmoPlacementAreaColor

```
[Tooltip("Specifies the gizmo color for placement area.")]
[SerializeField]
Color gizmoPlacementAreaColor
```

gizmoPlacementBoundsColor

```
[Tooltip("Specifies the gizmo color for placing object's bounds.")]
[SerializeField]
Color gizmoPlacementBoundsColor
```

gizmoPlacementGroundPositionsColor

```
[Tooltip("Specifies the gizmo color for positions on ground under the
placing object. " +
"These are indicators of where the ground position was
found for each edge of the bounds of the placing object.")]
[SerializeField]
Color gizmoPlacementGroundPositionsColor
```

AdditionalValidator

```
public IValidatePlacement AdditionalValidator
```

Validator invoked after internal validation has been performed with a valid result, if result is invalid this will not be invoked. Only already valid placements can be rejected by this validator.

CursorPositionProvider

```
public ICursorPositionProvider CursorPositionProvider
```

Cursor position provider reference. By default the DefaultMousePositionProvider is used. You can override this with any custom implementation of the ICursorPositionProvider interface.

isPlacementActive

```
private bool isPlacementActive
```

isPlacementValid

```
private bool isPlacementValid
```

targetPosition

```
private Vector3 targetPosition
```

targetRotationY

```
private float targetRotationY
```

currentRotationY

```
private float currentRotationY
```

placingObjectTemplate

```
private Transform placingObjectTemplate
```

placingObjectBounds

```
private IPlacementBounds placingObjectBounds
```

colliderHits

```
private readonly Collider[] colliderHits
```

PlacementPrefabs

```
public PlacementPrefabs PlacementPrefabs
```

Returns placement prefab component if present. This must be enabled on ObjectPlacement component in Editor under "Prefabs" tab, otherwise null will be returned.

Properties

Instance

```
public static ObjectPlacement Instance
     { get;
     }
```

Instance of currently active object placement. If second instance attempts to be instantiated, it will be destroyed on Awake.

PlacementArea

```
public PlacementAreaInfo PlacementArea
     { get;
     set; }
```

Specifies the area of valid placement. This can restrict placement of objects outside of certain area.

PlacementObjectPrefab

```
public GameObject PlacementObjectPrefab
     { get;
     }
```

Current prefab used when placement activates.

PlayerCamera

```
public Camera PlayerCamera
     { get;
     set; }
```

Sets and gets the current players camera used to render world objects. This camera is used for raycasts from screenpoint.

Methods

CurrentRotation

```
private Quaternion CurrentRotation()
```

IsPlacementActive

```
public bool IsPlacementActive()
```

Getter for checking if object placement is currently active.

OnValidate

```
private void OnValidate()
```

Awake

```
private void Awake()
```

Start

```
private void Start()
```

Update

```
private void Update()
```

Enable

```
public void Enable()
```

Activates the game object.

Disable

```
public void Disable()
```

Deactivates the game object.

SetPlacementPrefab

```
public void SetPlacementPrefab(
     GameObject prefab)
```

Sets the new placement prefab. If placement is already active it will be cancelled and activated again with the new prefab.

prefab: New placement prefab

BeginPlacement

```
public void BeginPlacement(
     GameObject prefab)
```

Begins the placement with the new prefab game object. Placement materials remain unchanged.

prefab: New placement prefab

BeginPlacement

```
public void BeginPlacement(
     GameObject prefab,
     Material[] validPlacement,
     Material[] invalidPlacement)
```

Begins placement with passed prefab and optional materials. If visuals on placing objects are desired, you can use placement material arguments, IPlacingObject or PlacingObjectObserver to apply visuals for placement states out of the box.

prefab: Prefab reference to be used for placement
validPlacement: Materials applied to new instance of the prefab when placement is valid. Default value is null, which does nothing.
invalidPlacement: Materials applied to new instance of the prefab when placement is invalid. Default value is null, which does nothing.

BeginPlacement

```
public void BeginPlacement()
```

Begins placement with current PlacementObjectPrefab on current's mouse position.

CancelPlacement

```
public void CancelPlacement()
```

Cancels currently active placement.

TryEndPlacement

```
public bool TryEndPlacement(
     out GameObject newInstance,
     bool removeDestructables = true)
```

Attempts to end placement by instantiating a new object on the valid placement. If placement failed no object will be instantiated and placement will continue and not be cancelled.

newInstance: Reference to the new instance created
removeDestructables: Specifies if destructables will be removed when placement ends. By default this is true

Returns: Returns true if new instance was created, returns false if placement was invalid.

ShouldIgnorePlacement

```
private bool ShouldIgnorePlacement()
```

Checks if placement should be ignored duo to cursor being over the UI components, like buttons, texts or images.

Returns: Returns true when placement should be ignored.

TryEndPlacement

```
public bool TryEndPlacement(
     out PlacementResult placementResult,
     bool removeDestructables = true)
```

Attempts to end placement by returning a valid position and rotation for the object currently placing. When placement is inactive or invalid false is returned along with default result values.

placementResult: Result values, set only in case of valid placement
removeDestructables: Specifies if destructables will be removed when placement ends. By default this is true

Returns: Returns true if placement was valid, returns false if it was inactive or invalid.

InternalTryEndPlacement

```
private bool InternalTryEndPlacement(
     out PlacementResult placementResult,
     bool removeDestructables)
```

Method for internal use, does not invoke any events on ObjectPlacementEvents, but only performs common logic for both interfaces to end placement. Attempts to end currently active placement and destroys placingObjectTemplate if placement is successful.

placementResult: Result of placement. Default values are returned if placement failed.

Returns: Returns true if placement was valid, returns false if it was inactive or invalid.

ResetRotation

```
public void ResetRotation()
```

Resets current rotation for placing object. This can also be used if preserveRotation is enabled.

RotateObjectYAxis

```
public void RotateObjectYAxis(
     float yAngle)
```

Rotates currently placing object on Y axis by increasing current target Y angle.

yAngle: Y angle to add to object’s current Y.

RotatePlacement

```
public void RotatePlacement(
     float directionalMagnitude)
```

Rotates currently placing object on Y axis by multiplying directional magnitude parameter with rotationSpeed.

directionalMagnitude: Y direction with magnitude.

UpdatePlacingObject

```
private void UpdatePlacingObject(
     bool animate)
```

Calculates placement position and rotation for the placing object, while validating and if needed updating renderer materials.

animate: If update should be animated. Applies to position and rotation changes.

ValidatePlacement

```
private bool ValidatePlacement(
     IPlacementBounds placingObject)
```

Validates placement bounds with ground alignment and bounds collision check.

placingObject: Placing objects bounds

Returns: Returns true if placement positon & rotation is valid.

TryGroundAlignment

```
private bool TryGroundAlignment(
     Bounds bounds,
     Quaternion rotation)
```

Performs ground alignment based on groundAlignmentType. When the type is GroundAlignmentType.ValidateOnly object will not be adjusted in position or rotation, but will still be validated based on ground angle threshold.

bounds: World space bounds
rotation: World rotation

Returns: Returns true if the placement is within expected groundAngleThreshold.

CheckForCollisions

```
private int CheckForCollisions(
     Bounds bounds,
     Quaternion rotation,
     LayerMask layer)
```

Performs Physics overlapping collision check with specified bounds and rotation.

bounds: World space bounds
rotation: World rotation
layer: Layer for collision

Returns: Returns true if there are NO colliders in the way.

UpdatePlacementMaterials

```
private void UpdatePlacementMaterials(
     Transform placingObject,
     bool isValid)
```

Updates materials on all child Renderer components. In case no materials are set, this does nothing.

placingObject: Root transform of the object
isValid: If placement is valid

GetPlacementPosition

```
private Vector3 GetPlacementPosition()
```

Shoots raycast based on current cursor position, validates the hit point and returns a validated position (snap and clamp features). In case of failure the last valid position is returned.

Returns: Returns validated position for the placement of an object.

GetPlacingObjectComponent

```
private IPlacingObject GetPlacingObjectComponent(
     Transform transform)
```

Gets a component that implements IPlacingObject.

RemoveDestructables

```
private void RemoveDestructables(
     Bounds bounds,
     Quaternion rotation)
```

Removes objects with colliders within the specified bounds and rotation.

bounds: Bounds of placed object.
rotation: Rotation of placed object.

OnDrawGizmos

```
private void OnDrawGizmos()
```

ObjectPlacementEditor.cs

Namespaces

TRavljen.PlacementSystem.Editor

```
namespace TRavljen.PlacementSystem.Editor
```

Classes

ObjectPlacementEditor

```
[CustomEditor(typeof(ObjectPlacement))]
public class ObjectPlacementEditor
: Editor
```

Variables

placementObjectPrefab

```
private SerializedProperty placementObjectPrefab
```

groundLayer

```
private SerializedProperty groundLayer
```

obstacleLayer

```
private SerializedProperty obstacleLayer
```

destructablesLayer

```
private SerializedProperty destructablesLayer
```

destroyOnPlacement

```
private SerializedProperty destroyOnPlacement
```

maxDetectionDistance

```
private SerializedProperty maxDetectionDistance
```

ignoreWhenOverUI

```
private SerializedProperty ignoreWhenOverUI
```

playerCamera

```
private SerializedProperty playerCamera
```

rotationSpeed

```
private SerializedProperty rotationSpeed
```

preserveRotation

```
private SerializedProperty preserveRotation
```

placementAreaType

```
private SerializedProperty placementAreaType
```

placementAreaBox

```
private SerializedProperty placementAreaBox
```

placementAreaSphere

```
private SerializedProperty placementAreaSphere
```

clampInsidePlacementArea

```
private SerializedProperty clampInsidePlacementArea
```

groundAlignmentType

```
private SerializedProperty groundAlignmentType
```

groundAlignmentDetectionOffset

```
private SerializedProperty groundAlignmentDetectionOffset
```

groundAngleThreshold

```
private SerializedProperty groundAngleThreshold
```

usePositionSnapping

```
private SerializedProperty usePositionSnapping
```

gridOrigin

```
private SerializedProperty gridOrigin
```

gridScale

```
private SerializedProperty gridScale
```

applyPlacementMaterials

```
private SerializedProperty applyPlacementMaterials
```

validPlacementMaterials

```
private SerializedProperty validPlacementMaterials
```

invalidPlacementMaterials

```
private SerializedProperty invalidPlacementMaterials
```

allowPlacementDuringAnimations

```
private SerializedProperty allowPlacementDuringAnimations
```

animatingAngleThreshold

```
private SerializedProperty animatingAngleThreshold
```

animatingDistanceThreshold

```
private SerializedProperty animatingDistanceThreshold
```

animatePositionChange

```
private SerializedProperty animatePositionChange
```

positionChangeSpeed

```
private SerializedProperty positionChangeSpeed
```

animateRotationChange

```
private SerializedProperty animateRotationChange
```

rotationChangeSpeed

```
private SerializedProperty rotationChangeSpeed
```

gizmoPlacementAreaColor

```
private SerializedProperty gizmoPlacementAreaColor
```

gizmoPlacementBoundsColor

```
private SerializedProperty gizmoPlacementBoundsColor
```

gizmoPlacementGroundPositionsColor

```
private SerializedProperty gizmoPlacementGroundPositionsColor
```

gizmoGridOriginColor

```
private SerializedProperty gizmoGridOriginColor
```

gizmoGridPointsColor

```
private SerializedProperty gizmoGridPointsColor
```

gizmoGridPlacementPointColor

```
private SerializedProperty gizmoGridPlacementPointColor
```

gizmosGridSphereSize

```
private SerializedProperty gizmosGridSphereSize
```

debugLogsEnabled

```
private SerializedProperty debugLogsEnabled
```

drawGizmosEnabled

```
private SerializedProperty drawGizmosEnabled
```

objectPlacement

```
private ObjectPlacement objectPlacement
```

inputSystemEditor

```
private Editor inputSystemEditor
```

observerEditor

```
private Editor observerEditor
```

prefabsEditor

```
private Editor prefabsEditor
```

selectedTabKey

```
private const string selectedTabKey
```

Properties

SelectedTabIndex

```
private int SelectedTabIndex
     { set;
     get; }
```

Methods

OnEnable

```
private void OnEnable()
```

OnDisable

```
private void OnDisable()
```

OnInspectorGUI

```
public override void OnInspectorGUI()
```

SetupTabGUI

```
private void SetupTabGUI()
```

VisualsTabGUI

```
private void VisualsTabGUI()
```

InputTabGUI

```
private void InputTabGUI()
```

PrefabTabsGUI

```
private void PrefabTabsGUI()
```

EventsTabGUI

```
private void EventsTabGUI()
```

RenderPlacementAreaInfo

```
private void RenderPlacementAreaInfo()
```

RightAlignedButton

```
private static bool RightAlignedButton(
     string text)
```

AddChildObjectComponent

```
private ComponentType AddChildObjectComponent(
     string name)
```

DestroyScript

```
private void DestroyScript(
     MonoBehaviour script,
     ref Editor scriptEditor)
```

RenderRemoveComponentGUI

```
private void RenderRemoveComponentGUI(
     string buttonText,
     MonoBehaviour script,
     ref Editor customEditor)
```

SetDirty

```
private void SetDirty(
     Object target)
```

IsObjectInvalid

```
private static bool IsObjectInvalid(
     GameObject go)
```

Bounds

APlacementBounds.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

APlacementBounds

```
public abstract class APlacementBounds
: MonoBehaviour, IPlacementBounds
```

Abstract MonoBehaviour for implementing IPlacementBounds. Subclass this component to achieve custom bounds behaviour.

Variables

drawGizmo

```
[SerializeField]
private bool drawGizmo
```

Properties

PlacementBounds

```
public abstract Bounds PlacementBounds
     { get;
     }
```

PlacementRotation

```
public abstract Quaternion PlacementRotation
     { get;
     }
```

Methods

OnDrawGizmos

```
private void OnDrawGizmos()
```

ColliderPlacementBounds.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

ColliderPlacementBounds

```
public class ColliderPlacementBounds
: APlacementBounds
```

Defines placement bounds with use of BoxCollider. When placing an object should consider bounds of a box collider, use this component. It is using manual calculation of the collider bounds because when collider is disabled (for any reason) those bounds (collder.bounds) will be invalid and cannot be used for placement bounds.

Variables

placementCollider

```
[Tooltip("Specifies the collider for defining object’s placement
bounds.")]
[SerializeField]
private BoxCollider placementCollider
```

bounds

```
[SerializeField, HideInInspector]
private Bounds bounds
```

Methods

PlacementBounds

```
public override Bounds PlacementBounds()
```

PlacementRotation

```
public override Quaternion PlacementRotation()
```

OnValidate

```
private void OnValidate()
```

UpdateBounds

```
private void UpdateBounds()
```

SetCollider

```
public void SetCollider(
     BoxCollider collider)
```

CustomPlacementBounds.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

CustomPlacementBounds

```
public class CustomPlacementBounds
: APlacementBounds
```

Defines custom bounds for the game object, which is required for placing an object using ObjectPlacement.

Variables

center

```
[Tooltip("Specifies center (offset) of the custom bounds in
local space.")]
public Vector3 center
```

size

```
[Tooltip("Specifies size of the custom bounds.")]
public Vector3 size
```

_transform

```
private Transform _transform
```

Methods

PlacementBounds

```
public override Bounds PlacementBounds()
```

PlacementRotation

```
public override Quaternion PlacementRotation()
```

Awake

```
private void Awake()
```

OnValidate

```
private void OnValidate()
```

IPlacementBounds.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

IPlacementBounds

```
public interface IPlacementBounds
```

Interface for providing placement bounds for a placing object.

Properties

gameObject

```
public GameObject gameObject
     { get;
     }
```

Root game object of the placing object.

PlacementBounds

```
public Bounds PlacementBounds
     { get;
     }
```

Bounds relative to local position of the root gameObject.

PlacementRotation

```
public Quaternion PlacementRotation
     { get;
     }
```

Absolute rotation of the placing object.

RendererPlacementBounds.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

RendererPlacementBounds

```
public class RendererPlacementBounds
: APlacementBounds
```

Defines placement bounds with use of Renderer. When placing an object should consider bounds of a renderer, use this component.

Variables

_renderer

```
[Tooltip("Specifies the renderer used to extract bounds from")]
[SerializeField]
private Renderer _renderer
```

Methods

PlacementBounds

```
public override Bounds PlacementBounds()
```

PlacementRotation

```
public override Quaternion PlacementRotation()
```

SetRenderer

```
public void SetRenderer(
     Renderer renderer)
```

Sets new renderer for placement bounds. Null should not be passed, instead disable or destroy this component.

renderer: New renderer

Input

AInputControl.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

AInputControl

```
[System.Serializable]
public abstract class AInputControl
: MonoBehaviour
```

Abstract class for defining player’s interface with placement system. It is intendede that this component listens to player input for specific input packages, while InputControlResponder handles interactions with the system.

Properties

OnPlacementCancel

```
public UnityEvent OnPlacementCancel
     { get;
     set; }
```

Invoke this when action for canceling selection was triggered.

OnPlacementActiveToggle

```
public UnityEvent OnPlacementActiveToggle
     { get;
     set; }
```

Invoke this once mouse down action is true (mouse click).

OnPlacementFinish

```
public UnityEvent OnPlacementFinish
     { get;
     set; }
```

Invoke this once mouse up action is true (mouse released).

OnRotate

```
public UnityEvent<float> OnRotate
     { get;
     set; }
```

Invoke this with direction and magnitude of the rotation, positive is clockwise and negative is counter clockwise.

OnNextPrefab

```
public UnityEvent OnNextPrefab
     { get;
     set; }
```

Invoke this when placement prefab should be changed to the next one.

OnPreviousPrefab

```
public UnityEvent OnPreviousPrefab
     { get;
     set; }
```

Invoke this when placement prefab should be changed to the previous one.

OnPrefabAction

```
public UnityEvent<int> OnPrefabAction
     { get;
     set; }
```

Invoke this when placement prefab should be changed for one on specific index.

ActionInputControl.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

ActionInputControlEditor

```
[CustomEditor(typeof(ActionInputControl))]
public class ActionInputControlEditor
: HiddenScriptPropertyEditor
```

ActionInputControl

```
public class ActionInputControl
: AInputControl, ICursorPositionProvider
```

Behaviour component for defining players input by using Unity’s new UnityEngine.InputSystem. This component is only available when the package for New Input system is in the project.

Enumerations

CursorPositionType

```
enum CursorPositionType{
     Mouse,
     TouchScreen}
```

Mouse: Uses current position of the mouse cursor.

TouchScreen: Uses current touch position on screen.

Variables

cancelAction

```
[Header("General")]
[Tooltip("Specifies the action used for canceling active
placement.")]
[Space]
public InputAction cancelAction
```

toggleActivePlacementAction

```
[Space]
[Tooltip("Specifies the action for starting placement process
with " +
"currently selected game object prefab.")]
public InputAction toggleActivePlacementAction
```

finishPlacementAction

```
[Space]
[Tooltip("Specifies the action for finishing placement process "
+
"(placing object in world).")]
public InputAction finishPlacementAction
```

rotateClockwiseAction

```
[Header("Rotation")]
[Space]
[Tooltip("Specifies the clockwise rotation input action. You can
change " +
"rotation behaviour from <b>Press</b> to <b>Hold</b>
by changing the " +
"Action Type from <b>Button</b> to
<b>Passthrough</b> or <b>Value</b>.")]
public InputAction rotateClockwiseAction
```

rotateCounterClockwiseAction

```
[SerializeField, Space]
[Tooltip("Specifies the counter clockwise rotation input action.
" +
"You can change rotation behaviour from <b>Press</b>
to <b>Hold</b> by " +
"changing the Action Type from <b>Button</b> to
<b>Passthrough</b> or " +
"<b>Value</b>.")]
public InputAction rotateCounterClockwiseAction
```

nextPrefabAction

```
[Header("Prefab Control")]
[Space]
public InputAction nextPrefabAction
```

previousPrefabAction

```
public InputAction previousPrefabAction
```

prefabActions

```
[Space]
public InputAction[] prefabActions
```

cursorPositionType

```
[Tooltip("Specifies the type of cursor position behaviour. By
default " +
"mouse position is read, alternatively touch screen
position can be " +
"used for mobile devices and such.")]
[SerializeField, Header("Cursor")]
private CursorPositionType cursorPositionType
```

_cursorPositionType

```
[SerializeField, HideInInspector]
private CursorPositionType _cursorPositionType
```

cursorPositionAction

```
[Tooltip("Specifies the cursor position. Most commonly this
would be " +
"mouse position or touch screen position.
Alternatively custom one " +
"can be added for others like controller sticks.")]
[Space]
public InputAction cursorPositionAction
```

_initialSetup

```
[SerializeField, HideInInspector]
private bool _initialSetup
```

Properties

CursorPosition

```
public Vector3 CursorPosition
     { get;
     }
```

Methods

OnEnable

```
private void OnEnable()
```

OnDisable

```
private void OnDisable()
```

Start

```
private void Start()
```

Update

```
private void Update()
```

OnValidate

```
private void OnValidate()
```

OnDestroy

```
private void OnDestroy()
```

HandleCancelAction

```
private void HandleCancelAction(
     InputAction.CallbackContext context)
```

HandleToggleActivePlacementAction

```
private void HandleToggleActivePlacementAction(
     InputAction.CallbackContext context)
```

HandleFinishPlacementAction

```
private void HandleFinishPlacementAction(
     InputAction.CallbackContext context)
```

HandleClockwiseRotation

```
private void HandleClockwiseRotation(
     InputAction.CallbackContext context)
```

HandleCounterClockwiseRotation

```
private void HandleCounterClockwiseRotation(
     InputAction.CallbackContext context)
```

HandleNextPrefabAction

```
private void HandleNextPrefabAction(
     InputAction.CallbackContext context)
```

HandlePreviousPrefabAction

```
private void HandlePreviousPrefabAction(
     InputAction.CallbackContext context)
```

HandlePrefabAction

```
private void HandlePrefabAction(
     InputAction.CallbackContext context)
```

InitialSetupIfNeeded

```
private void InitialSetupIfNeeded()
```

SetupCursorAction

```
private void SetupCursorAction()
```

IsPressed

```
private bool IsPressed(
     InputAction.CallbackContext context)
```

IsPressed

```
private bool IsPressed(
     InputAction action)
```

InputControlResponder.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

InputControlResponder

```
internal sealed class InputControlResponder
: MonoBehaviour
```

Responder class for invoking methods on ObjectPlacement based on AInputControl component events which must be attached to the same game object.

Variables

placement

```
private ObjectPlacement placement
```

Methods

Awake

```
private void Awake()
```

OnEnable

```
private void OnEnable()
```

OnDisable

```
private void OnDisable()
```

AddListeners

```
private void AddListeners()
```

RemoveListeners

```
private void RemoveListeners()
```

ToggleActivePlacement

```
private void ToggleActivePlacement()
```

CancelPlacement

```
private void CancelPlacement()
```

EndPlacement

```
private void EndPlacement()
```

RotatePlacement

```
private void RotatePlacement(
     float directionalMagnitude)
```

KeyInputControl.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

KeyInputControlEditor

```
[CustomEditor(typeof(KeyInputControl))]
public class KeyInputControlEditor
: HiddenScriptPropertyEditor
```

KeyInputControl

```
[System.Serializable]
public class KeyInputControl
: AInputControl
```

Behaviour component for defining players input by using Unity’s OLD Input.

Variables

toggleActivePlacementKey

```
[Header("General")]
[Tooltip("Specifies the key for starting placement process with
" +
"currently selected game object prefab.")]
public KeyCode toggleActivePlacementKey
```

cancelPlacementKey

```
[Tooltip("Specifies the key used for canceling active
placement.")]
public KeyCode cancelPlacementKey
```

finishPlacementKey

```
[Tooltip("Specifies the key for finishing placement process " +
"(placing object in world).")]
public KeyCode finishPlacementKey
```

holdToRotate

```
[Header("Rotation")]
[Tooltip("Specifies if rotation should be applied on key hold or
key press.")]
public bool holdToRotate
```

rotateClockwiseKey

```
[Tooltip("Specifies the clockwise rotation input key.")]
public KeyCode rotateClockwiseKey
```

rotateCounterClockwiseKey

```
[Tooltip("Specifies the counter clockwise rotation input key.")]
public KeyCode rotateCounterClockwiseKey
```

nextPrefabKey

```
public KeyCode nextPrefabKey
```

previousPrefabKey

```
public KeyCode previousPrefabKey
```

prefabKeys

```
public KeyCode[] prefabKeys
```

Cursor

DefaultMousePositionProvider.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

DefaultMousePositionProvider

```
sealed class DefaultMousePositionProvider
: ICursorPositionProvider
```

Internal implementation for default mouse position behaviour. ActionInputControl implements the ICursorPositionProvider to add touch screen support.

Properties

CursorPosition

```
public Vector3 CursorPosition
     { get;
     }
```

ICursorPositionProvider.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

ICursorPositionProvider

```
public interface ICursorPositionProvider
```

Interface for implementing a custom cursor position handling. It can be used for mouse, touch screen and even controller sticks.

Properties

CursorPosition

```
public Vector3 CursorPosition
     { get;
     }
```

Current cursor position, most commonly this would be current mouse position.

Interface

IPlacingObject.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

IPlacingObject

```
public interface IPlacingObject
```

Interface for notifying a specific game object of its placement status. Implement this interface by subclassing a MonoBehaviour component and then update your game object with custom behaviour upon these events.

Methods

PlacementStarted

```
public void PlacementStarted()
```

Invoked when placement on this object has started. In such case the object is a template instance created for active placement only.

PlacementValidated

```
public void PlacementValidated(
     bool isValid)
```

Invoked when placement on this object is validated.

isValid: If placement of the object is valid at this moment.

PlacementEnded

```
public void PlacementEnded()
```

Invoked when placement of this object is finished. This will be invoked on final instance created when placement is complete.

PlacementFailed

```
public void PlacementFailed()
```

Invoked when placement of this object has failed. This happens when placement is attempted with invalid position and rotation.

PlacementCancelled

```
public void PlacementCancelled()
```

Invoked when placement of this object is cancelled.

IValidatePlacement.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

IValidatePlacement

```
public interface IValidatePlacement
```

Interface for additional custom validation for object’s placement. You can implement this to verify if certain objects can be placed in certain locations, under specific rotation.

Methods

IsPlacementValid

```
public bool IsPlacementValid(
     GameObject gameObject,
     Bounds bounds,
     Quaternion rotation)
```

Validates placement of a game object.

gameObject: Game object to validate
bounds: Bounds of the game object
rotation: Rotation of bounds in world space

Returns: Returns true if object has been validated and can be placed, returns false when it cannot be placed.

ObjectPlacementEvents.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

ObjectPlacementEvents

```
sealed public class ObjectPlacementEvents
```

Class holding references to public events invoked when placement system is active, respectively.

Variables

Instance

```
public static readonly ObjectPlacementEvents Instance
```

OnPlacementStart

```
public readonly UnityEvent<GameObject> OnPlacementStart
```

Event invoked when placement has started, receiving the object instance created for placement.

OnPlacementValidate

```
public readonly UnityEvent<bool> OnPlacementValidate
```

Event invoked when placement of an object is finished validating.

OnPlacementEnd

```
public readonly UnityEvent<GameObject,PlacementResult>
OnPlacementEnd
```

Event invoked when placement has ended successfully, receiving the new game object instance and the result of placement. In case ObjectPlacement.TryEndPlacement(out PlacementResult) is invoked, then the new object received will be null.

OnPlacementFail

```
public readonly UnityEvent OnPlacementFail
```

Event invoked when placement has been attempted, placement validation prevented the placement.

OnPlacementCancel

```
public readonly UnityEvent OnPlacementCancel
```

Event invoked when an active placement has been canceled.

OnObjectDestroy

```
public readonly UnityEvent<GameObject> OnObjectDestroy
```

Event invoked when an object will or should be destroyed. Depending on placement configuration.

ObjectPlacementObserver.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

ObjectPlacementObserverEditor

```
[CustomEditor(typeof(ObjectPlacementObserver))]
public class ObjectPlacementObserverEditor
: HiddenScriptPropertyEditor
```

ObjectPlacementObserver

```
public class ObjectPlacementObserver
: MonoBehaviour
```

MonoBehaviour component used for observing all the events on ObjectChangeEvents and expose them in Unity Editor. This is a convenience script that can be used in Editor instead of adding listeners through the code.

Variables

OnPlacementStart

```
[Tooltip("Event invoked when placement has begun, receiving the
object instance created for placement.")]
public UnityEvent<GameObject> OnPlacementStart
```

OnPlacementValidate

```
[Tooltip("Event invoked when placement of an object is finished
validating.")]
public UnityEvent<bool> OnPlacementValidate
```

OnPlacementEnd

```
[Tooltip("Event invoked when placement has ended successfully,
receiving the new game object instance and the result of
placement.\n" +
"In case 'ObjectPlacement.TryEndPlacement(out
PlacementResult)' is invoked, then the new object received will
be null.")]
public UnityEvent<GameObject,PlacementResult> OnPlacementEnd
```

OnPlacementFail

```
[Tooltip("Event invoked when placement has been attempted,
placement validation prevented the placement.")]
public UnityEvent OnPlacementFail
```

OnPlacementCancel

```
[Tooltip("Event invoked when an active placement has been
canceled.")]
public UnityEvent OnPlacementCancel
```

OnObjectDestroy

```
[Tooltip("Event invoked when an object will or should be
destroyed. Depending on placement configuration.")]
public UnityEvent<GameObject> OnObjectDestroy
```

events

```
private ObjectPlacementEvents events
```

Methods

OnEnable

```
private void OnEnable()
```

OnDisable

```
private void OnDisable()
```

PlacementPrefabs.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

PlacementPrefabs

```
public class PlacementPrefabs
: MonoBehaviour
```

Manages the selection and placement of prefabs within the scene. This class handles cycling through a list of prefabs and updating the active prefab for placement.

Variables

prefabs

```
[Tooltip("Array of prefabs that can be cycled through for
placement.")]
[SerializeField]
private GameObject[] prefabs
```

cyclesRotation

```
[Tooltip("Specifies whether the prefab selection will cycle
(wrap around) when reaching the start or end of the list.")]
[SerializeField]
private bool cyclesRotation
```

index

```
private int index
```

Index of the currently selected prefab.

Methods

Prefabs

```
public GameObject[] Prefabs()
```

Gets the array of prefabs that can be selected for placement.

Start

```
private void Start()
```

Initializes the component by setting up the input control listeners.

SetPrefabs

```
public void SetPrefabs(
     GameObject[] prefabs)
```

Sets the array of prefabs that can be selected for placement.

prefabs: Array of prefabs to assign.

Next

```
public void Next()
```

Selects the next prefab in the array. If cycling is enabled, wraps around to the first prefab.

Back

```
public void Back()
```

Selects the previous prefab in the array. If cycling is enabled, wraps around to the last prefab.

SetPrefabIndex

```
public void SetPrefabIndex(
     int index)
```

Sets the prefab index to the specified value if within valid range.

index: The index of the prefab to select.

ClampIndex

```
private int ClampIndex(
     int newIndex)
```

Clamps the index to ensure it remains within the valid range, optionally cycling if enabled.

newIndex: The new index to clamp.

Returns: The clamped index.

UpdateIndex

```
private void UpdateIndex(
      int newIndex)
```

Updates the index of the selected prefab and sets the active placement prefab in the ObjectPlacement system.

newIndex: The new index to set.

PlacementPrefabsEditor.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

PlacementPrefabsEditor

```
[CustomEditor(typeof(PlacementPrefabs))]
class PlacementPrefabsEditor
: UnityEditor.Editor
```

Variables

prefabs

```
SerializedProperty prefabs
```

cyclesRotation

```
SerializedProperty cyclesRotation
```

Methods

OnEnable

```
private void OnEnable()
```

OnInspectorGUI

```
public override void OnInspectorGUI()
```

PlacementResult.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

PlacementResult

```
public struct PlacementResult
```

Information about the placement result. It contains objects placing position and it’s rotation.

Variables

Position

```
public Vector3 Position
```

Position of the placement.

Rotation

```
public Quaternion Rotation
```

Rotation of the placement.

PlacingObjectObserver.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

PlacingObjectObserver

```
public class PlacingObjectObserver
: MonoBehaviour, IPlacingObject
```

Observer component that implements IPlacingObject and supports referencing events within the Unity Editor. This is for cases where a simple action needs to be performed based on an event to avoid implementing a custom IPlacingObject MonoBehaviour.

Variables

OnPlacementStart

```
[Tooltip("Invoked when placement on this object has started. " +

"In such case the object is a template instance
created for active placement only.")]
public UnityEvent OnPlacementStart
```

OnPlacementValidated

```
[Tooltip("Invoked when placement on this object is validated.\n"
+
"Value passed is true if placement of the object is
valid at this moment.")]
public UnityEvent<bool> OnPlacementValidated
```

OnPlacementEnd

```
[Tooltip("Invoked when placement of this object is finished. " +

"This will be invoked on final instance created when
placement is complete.")]
public UnityEvent OnPlacementEnd
```

OnPlacementCancel

```
[Tooltip("Invoked when placement of this object is cancelled.")]
public UnityEvent OnPlacementCancel
```

OnPlacementFail

```
[Tooltip("Invoked when placement of this object has failed. " +
"This happens when placement is attempted with
invalid position and rotation.")]
public UnityEvent OnPlacementFail
```

Methods

PlacementStarted

```
public void PlacementStarted()
```

PlacementValidated

```
public void PlacementValidated(
     bool isValid)
```

PlacementEnded

```
public void PlacementEnded()
```

PlacementCancelled

```
public void PlacementCancelled()
```

PlacementFailed

```
public void PlacementFailed()
```

PlacementArea

ASpherePlacementAreas.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

ASpherePlacementAreas

```
[RequireComponent(typeof(ObjectPlacement))]
public abstract class ASpherePlacementAreas
: MonoBehaviour, IPlacementArea
```

Variables

placement

```
[HideInInspector]
[SerializeField]
protected ObjectPlacement placement
```

overrideType

```
[Tooltip("Specifies if the boundsType should be overridden.
Leave this disabled and " +
"manually control the type of the placement
area.")]
[SerializeField]
protected bool overrideType
```

gizmosColor

```
[Header("Preview")]
[SerializeField]
protected Color gizmosColor
```

alwaysShowGizmos

```
[SerializeField]
protected bool alwaysShowGizmos
```

Methods

GetAreas

```
protected abstract SphereBounds[] GetAreas()
```

Awake

```
protected virtual void Awake()
```

OnValidate

```
protected virtual void OnValidate()
```

IPlacementArea.ClosestPosition

```
Vector3 IPlacementArea.ClosestPosition(
     Vector3 position)
```

IPlacementArea.IsInside

```
bool IPlacementArea.IsInside(
      Vector3 position)
```

OnDrawGizmos

```
protected virtual void OnDrawGizmos()
```

IPlacementArea.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

IPlacementArea

```
public interface IPlacementArea
```

Defines contract for a placement area.

Methods

ClosestPosition

```
public Vector3 ClosestPosition(
      Vector3 position)
```

Returns closest position within the area.

IsInside

```
public bool IsInside(
      Vector3 position)
```

Checks whether the position is inside the area.

ObjectPlacementArea.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

ObjectPlacementAreaEditor

```
[CustomEditor(typeof(ObjectPlacementArea), true)]
public sealed class ObjectPlacementAreaEditor
: HiddenScriptPropertyEditor
```

Methods

OnInspectorGUI

```
public override void OnInspectorGUI()
```

ObjectPlacementArea

```
[DisallowMultipleComponent]
public class ObjectPlacementArea
: MonoBehaviour
```

Component which specifies a valid placement area for a game object. This component manages adding and removing placement are on ObjectPlacementAreaManager when enabled or disabled. To manage when placement area is added, simply enable and disable this component.

Variables

areaBounds

```
[Tooltip("Specifies area bounds provided by the game object.
Position is relative.")]
[SerializeField]
private SphereBounds areaBounds
```

gizmoColor

```
[Header("Preview")]
[SerializeField]
private Color gizmoColor
```

disableGizmo

```
[SerializeField]
private bool disableGizmo
```

Properties

WorldBounds

```
public SphereBounds WorldBounds
     { get;
     }
```

Methods

Start

```
private void Start()
```

OnEnable

```
private void OnEnable()
```

OnDisable

```
private void OnDisable()
```

TryGetManager

```
private bool TryGetManager(
     out ObjectPlacementAreaManager manager)
```

OnDrawGizmos

```
private void OnDrawGizmos()
```

ObjectPlacementAreaManager.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

ObjectPlacementAreaManagerEditor

```
[CustomEditor(typeof(ObjectPlacementAreaManager), true)]
class ObjectPlacementAreaManagerEditor
: UnityEditor.Editor
```

Methods

OnInspectorGUI

```
public override void OnInspectorGUI()
```

ObjectPlacementAreaManager

```
public class ObjectPlacementAreaManager
: ASpherePlacementAreas
```

Component which manages a list of objects and their placement areas.

Variables

_objectAreas

```
private readonly Dictionary<GameObject,SphereBounds>
_objectAreas
```

areas

```
private SphereBounds[] areas
```

Methods

GetAreas

```
protected override SphereBounds[] GetAreas()
```

Awake

```
protected override void Awake()
```

AddObjectArea

```
public void AddObjectArea(
     GameObject obj,
     Vector3 position,
     float radius)
```

Add area for the game object.

obj: Area game object owner
position: Center world position.
radius: Radius

RemoveObjectArea

```
public void RemoveObjectArea(
     GameObject obj)
```

Remove area for the game object.

obj: Area game object owner.

OnDrawGizmos

```
protected override void OnDrawGizmos()
```

PlacementAreaInfo.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Enumerations

PlacementAreaType

```
public enum PlacementAreaType{
     Anywhere,
     Box,
     Sphere,
     Custom}
```

Type of the bounds in which the objects can be placed.

Anywhere: Objects can be placed anywhere where there is ground.

Box: Objects can be placed on ground within the box bounds of the area.

Sphere: Objects can be placed on ground in the sphere bounds of the area.

Custom: Placement area defined by a custom implementation by implementing IPlacementArea interface.

Classes

PlacementAreaInfo

```
[Serializable]
public struct PlacementAreaInfo
: IPlacementArea
```

Specifies the information about valid placement area for objects. When placement area is limited, use ClosestPosition to limit the positions within the valid bounds.

Variables

boundsType

```
[Tooltip("Specifies placement area type. In cases where map has
a limit or " +
"placement is allowed around certain objects use
this to limit " +
"the placement area.")]
public PlacementAreaType boundsType
```

boxBounds

```
[Tooltip("Specifies placement area with box shape. Set it’s
position and size.")]
public Bounds boxBounds
```

sphereBounds

```
[Tooltip("Specifies placement area with sphere shape. Set it’s
position and size.")]
public SphereBounds sphereBounds
```

customArea

```
public IPlacementArea customArea
```

Specifies custom placement area component. This allows easy use of multiple placement areas with custom implementation.

Methods

ClosestPosition

```
public Vector3 ClosestPosition(
     Vector3 position)
```

Clamps position within the valid placement area. For PlacementAreaType.Anywhere the same value will be returned, for limiting types the closest position within the bounds will be found.

position: Current position

Returns: Valid position

IsInside

```
public bool IsInside(
     Vector3 position)
```

RenderGizmos

```
public void RenderGizmos(
     Color areaColor)
```

Renders gizmo for current placement area.

areaColor: Color for area gizmo.

SpherePlacementAreaManager.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

SpherePlacementAreaManager

```
public class SpherePlacementAreaManager
: ASpherePlacementAreas
```

Component which manages a list of areas of type sphere/ring. You can add or remove them dynamically in runtime.

Variables

areas

```
[Tooltip("Specifies ring areas for valid placements. At least
one must be present!")]
[SerializeField]
protected SphereBounds[] areas
```

Utility

ColliderBoundsHelper.cs

Namespaces

TRavljen.PlacementSystem.Utility

```
namespace TRavljen.PlacementSystem.Utility
```

Classes

ColliderBoundsHelper

```
static public class ColliderBoundsHelper
```

Methods

CalculateBounds

```
static public Bounds CalculateBounds(
     BoxCollider collider)
```

Retrieving collider bounds while the collider itself is disabled will return Vector3 of zero size. This method will calculate it from collider center and size, while applying its transform’s rotation and scale. The bounds position is still in local space.

collider: Collider to calculate the Bounds for

GridSnapper.cs

Namespaces

TRavljen.PlacementSystem.Utility

```
namespace TRavljen.PlacementSystem.Utility
```

Classes

GridSnapper

```
[System.Serializable]
sealed public class GridSnapper
```

Constructors

GridSnapper

```
public  GridSnapper(
     Vector3 gridOrigin,
     Vector2 gridScale)
```

Creates a grid of specified origin and size.

Variables

gridScale

```
[SerializeField]
[Tooltip("Specifies the grid size (space between snapping
points).")]
private Vector2 gridScale
```

gridOrigin

```
[SerializeField]
[Tooltip("Specifies the origin of the snapping grid. Y axis is
not used.")]
private Vector3 gridOrigin
```

gizmoGridOriginColor

```
[SerializeField]
[Tooltip("Specifies the color of the grid origin sphere
gizmo.")]
private Color gizmoGridOriginColor
```

gizmoGridPointsColor

```
[SerializeField]
[Tooltip("Specifies the color of the grid point spheres
gizmo.")]
private Color gizmoGridPointsColor
```

gizmoGridPlacementPointColor

```
[SerializeField]
[Tooltip("Specifies the point of placement on the grid. " +
"Grid gizmos will follow this point to avoid
rendering too many.")]
private Color gizmoGridPlacementPointColor
```

gizmosSphereSize

```
[SerializeField]
[Range(0.1F, 10F)]
[Tooltip("Specifies the sphere size for grid points.")]
private float gizmosSphereSize
```

MaxCellRowCount

```
private const float MaxCellRowCount
```

Max cell count for single row/column. If changed, this should always remain an even number.

lastNereastPoint

```
private Vector3 lastNereastPoint
```

Specifies the last point on grid that was calculated. Vector3.zero is the default value.

Properties

ClearOrigin

```
private Vector3 ClearOrigin
     { get;
     }
```

Clears Y axis from origin.

Methods

RenderDebugPoints

```
public void RenderDebugPoints()
```

Renders spheres of the grid points, used for snapping.

Validate

```
public void Validate()
```

Validates and clamps the gridScale.

SnapToGrid

```
public Vector3 SnapToGrid(
     Vector3 position)
```

Snaps a position to the nearest grid point.

GroundPlacement.cs

Namespaces

TRavljen.PlacementSystem.Utility

```
namespace TRavljen.PlacementSystem.Utility
```

Classes

GroundAlignment

```
[System.Serializable]
class GroundAlignment
```

Finds ground bottom points of the rotated bounds in world space and if needed, aligns the bounds to ground with new position and rotation.

Constructors

GroundAlignment

```
public  GroundAlignment()
```

Variables

GroundDetectionOffset

```
[HideInInspector]
[Range(-150, 150)]
[Tooltip("Specifies the Y offset of raycast detection for ground
position. " +
"Detected points on bounds edges define the angle of
object placement.")]
public float GroundDetectionOffset
```

debugLogsEnabled

```
public bool debugLogsEnabled
```

bottomPoints

```
private Vector3[] bottomPoints
```

hit

```
private RaycastHit hit
```

Methods

BottomPoints

```
public Vector3[] BottomPoints()
```

TryGetAlignedPositionWithGround

```
public bool TryGetAlignedPositionWithGround(
     Bounds bounds,
     Quaternion currentRotation,
     LayerMask groundLayer,
     out Quaternion newRotation,
     out Vector3 newPosition)
```

Attempts to find the new position and rotation for the placing bounds.

bounds: Bounds of placing object
currentRotation: Rotation of bounds in world space
groundLayer: Layer of ground to detect placement colliders (mesh or terrain).
newRotation: New rotation in world space, aligned to ground
newPosition: New position in world space, positioned on ground

Returns: Returns true if alignment was successful, returns false if it was not

FindGroundPoints

```
public bool FindGroundPoints(
     Bounds bounds,
     Quaternion rotation,
     LayerMask groundLayer,
     ref Vector3[] points)
```

Takes the bottom 4 edges of the rotated bounds and finds the nearest ground below it using Physics.Raycast(Vector3, Vector3, out RaycastHit, float, int).

bounds: Bounds of placing object
rotation: Rotation of bounds in world space
groundLayer: Layer of ground to detect placement colliders (mesh or terrain).
points: Points updated if ground was found

Returns: Returns true if ground for all 4 edges was found.

CalculatePlaneNormal

```
static Vector3 CalculatePlaneNormal(
     Vector3[] points)
```

Calculates planes normal vector.

GetBottomEdges

```
static void GetBottomEdges(
     Vector3 center,
     Vector3 size,
     Quaternion rotation,
     ref Vector3[] edges)
```

Calculates the bottom edges of a bounding box in world space.

InstantiatePlacementObject.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

InstantiatePlacementObject

```
public static class InstantiatePlacementObject
```

Helper methods for instantiatin a new object template used for visual placement.

Methods

TryInstantiate

```
public static GameObject TryInstantiate(
     GameObject prefab,
     Vector3 position,
     Quaternion rotation,
     Transform parent,
     out IPlacementBounds bounds)
```

TryCreatePlacementBounds

```
public static bool TryCreatePlacementBounds(
     GameObject newInstance,
     out IPlacementBounds placementBounds)
```

IntRange.cs

Namespaces

TRavljen.Utility

```
namespace TRavljen.Utility
```

Classes

IntRange

```
[System.Serializable]
public struct IntRange
```

Specifies an integer range by defining its lower and upper bounds with min and max values.

Constructors

IntRange

```
public  IntRange(
     int min,
     int max)
```

Variables

min

```
public int min
```

Minimal/lower bound of the integer range.

max

```
public int max
```

Maximal/upper bound of the integer range.

Methods

Clamp

```
public float Clamp(
     float value)
```

Clamps the value within the range.

value: Value to clamp

Returns: Returns a value within the Integer range

IsWithinBounds

```
public bool IsWithinBounds(
     float value)
```

Checks if the value is within the integer range/bounds. Min and max values are inclusive.

value: Value to check

Returns: Returns true if the value is within integer range.

IntRangeAttribute.cs

Namespaces

TRavljen.Utility

```
namespace TRavljen.Utility
```

Classes

IntRangeAttribute

```
public class IntRangeAttribute
: PropertyAttribute
```

Constructors

IntRangeAttribute

```
public  IntRangeAttribute(
     int min,
     int max)
```

Properties

Min

```
public int Min
     { get;
     }
```

Max

```
public int Max
     { get;
     }
```

TRavljen.Utility

```
namespace TRavljen.Utility
```

Classes

IntRangeAttributeDrawer

```
[CustomPropertyDrawer(typeof(IntRangeAttribute))]
public class IntRangeAttributeDrawer
: PropertyDrawer
```

Methods

OnGUI

```
public override void OnGUI(
     Rect position,
     SerializedProperty property,
     GUIContent label)
```

MathHelper.cs

Namespaces

TRavljen.PlacementSystem.Utility

```
namespace TRavljen.PlacementSystem.Utility
```

Classes

MathHelper

```
static class MathHelper
```

Methods

AngleDistance

```
public static float AngleDistance(
     float angle1,
     float angle2)
```

Calculates the distance between two angles.

WrapAngle

```
public static float WrapAngle(
     float angle)
```

Wraps the angle within a valid range for calculation (euler). This means the returned value will be within -180 and 180 range by modifying the value as required (+/- 360).

SphereBounds.cs

Namespaces

TRavljen.PlacementSystem

```
namespace TRavljen.PlacementSystem
```

Classes

SphereBounds

```
[System.Serializable]
public struct SphereBounds
```

Defines bounds of a sphere with center and radius.

Constructors

SphereBounds

```
public  SphereBounds(
     Vector3 center,
     float radius)
```

Variables

center

```
[Tooltip("Specifies bounds center.")]
public Vector3 center
```

radius

```
[Tooltip("Specifies bounds radius.")]
[Range(0.01f, 200f)]
public float radius
```

Methods

Contains

```
public bool Contains(
     Vector3 point)
```

Checks if a point is within the sphere.

Intersects

```
public bool Intersects(
     SphereBounds other)
```

Checks if another sphere intersects with this sphere.

ToBounds

```
public Bounds ToBounds()
```

Get a bounding box that contains this sphere

ClosestPoint

```
public Vector3 ClosestPoint(
     Vector3 point)
```

Finds the closest point on the sphere surface to the given point.