Technical Design Document:

Chunk Editor Enhancements

Client: BUFFALO BUFFALO

Mr.Jezz Team - Team Epic Awesome: Mpho Bonde, Zimo Wu

Project time: 1/8/2024 ~ 4/11/2024

Table of Content

Introduction and Overview

This document outlines the technical specifications for the enhancements to the Chunk Editor tool for the game Fresh Tracks by Buffalo Buffalo, a vital tool in their game development suite. Aimed at improving level design workflows, these enhancements introduce multi-selection capabilities, UI window minimization features, and a runtime Gizmo application. The intended audience for this document includes developers and stakeholders involved in the project.

Revision History

Version 1.0

Date: 04 April 2024

Description of Changes: Initial creation of the document.

Author: Zimo Wu & Mpho Thor Bonde

Agreed Deliverables

- A Runtime Gizmo with ability to Move, Rotate and scale
- The ability to click and drag to select multiple object and then manipulate them
- A way to get more space by Moving/resizing or minimizing UI Windows.

Current System Overview

The Chunk Editor, operates primarily through the ChunkDesigner_Scene script, affectionately known as CHUNKY among friends and foe, it boasts over 3,000 lines of code, CHUNKY's comprehensive set of features forms the backbone of our design process.
However, its size and complexity made direct integrations challenging, leading to the decision to encapsulate new features in separate classes, adhering to the principle of separation of concerns.

Project Scope and Changes

We aimed to enhance the Chunk Editor with three key features—multi-selection tools, UI minimization, and a runtime Gizmo—without bloating the already large codebase. Opting for development through separate classes, we avoided complicating CHUNKY further, aiming for clarity, easy testing, and future scalability. This approach resulted in five distinct classes, each encapsulating a new feature, and a few minor changes to CHUNKY.

Architectural Overview

Our architecture emphasizes modularity and clear separation of concerns. New functionalities are encapsulated in independent classes, facilitating straightforward integration with CHUNKY and ensuring each can be developed, tested, and debugged separately. This strategy minimizes disruption to the existing codebase and enhances maintainability and adaptability.

Class Relationship

See Appendix 1-4

Detailed Design overview

Modifications to ChunkDesigner_Scene

The script ChunkDesigner_Scene, Or CHUNKY as it is called among friends, is a massive monster of a 3000+ lines chunker of a script, its enormous size only overshadowed by its importance.

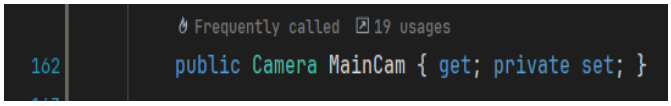
At first we Tried to gently put our code into this humongous beast, but quickly gave up in favor of a more separated approach. Therefor most code has been neatly put into own classes in order to abide by the perfectly sound rule of separation of concerns and what little changes to CHUNKY we had to make are as follows:

Key Modifications Overview

1. Public Access to Main Camera

Modification: Introduced an auto property MainCam with a private setter, initialized in the Start() method.

Purpose: Provides controlled access to Camera.Main, facilitating easier reference throughout the script and by external classes.



2. Highlighting Multiple Objects

Method Name: HighlightMultipleObjects(List<GameObject>)

Description: Accepts a list of GameObjects to apply highlighting effects. Clears existing highlights and applies new ones to all child renderers of each GameObject in the list.

Impact: Enhances visual feedback by allowing multiple objects to be highlighted simultaneously, improving usability in complex scenes.

3. Clearing Highlights

Method Name: ClearHighlight(List<GameObject>)

Description: Removes highlight effects from a list of GameObjects.

Impact: Complements HighlightMultipleObjects by providing a way to remove highlights, ensuring visual clarity and preventing unwanted persistence of highlight effects.

```
880     //************************ Gizmo and Dragging Methods
881
882     /// Highlights multiple GameObjects by adding their child renderers to the rendering list of an outline script. ...
        Frequently called  2 usages
891     public void HighlightMultipleObjects(List<GameObject> objectsToHighlight)
892     {
893         // Clear any previous highlights
894         outline.ClearRenderingList();
895
896         // Early out if there are no objects to highlight
897         if (objectsToHighlight == null || objectsToHighlight.Count == 0) return;
898
899         // Add all child renderers of each object in the list to the rendering list of the outline script
900         outline.AddAllChildRenderersToRenderingList(EPOOutline.RenderersAddingMode.MeshRenderer
901                                                     | EPOOutline.RenderersAddingMode.SkinnedMeshRenderer, objectsToHighlight);
902     }
903
904     /// Clears the highlight from the provided list of GameObjects and the rendering list of the outline script. ...
        Frequently called  1 usage
912     public void ClearHighlight(List<GameObject> objectsToHighlight)
913     {
914         objectsToHighlight.Clear();
915         outline.ClearRenderingList();
916     }
917
918     // ************************************************************
```

4. Hover Query Adjustment

Modification: Added a condition in HoverQuery() to prevent highlight updates when hovering over an object, maintaining the current highlight state.

Purpose: Preserves highlight integrity, particularly when multiple objects are selected, enhancing user interaction with the scene.

```
933     public void HoverQuery()
934     {
935         GameObject lastHoveredObject = currentHover.obj;
936         currentHover = new HoverQueryResult();
937         GetHoveredObject(currentHover);
938
939         if (currentHover.obj != lastHoveredObject)
940         {
941             if(DragSelectionManager.Instance.currentState != DragSelectionManager.DragSelectionInteractionState.MultipleObjectsSelected)
942             {
943                 UpdateHoverHighlight();
944             }
945         }
946     }
```

5. Deletion Mode Expansion

Method Name: SetDeleteMode()

Description: Expanded to disable any active gizmos when entering deletion mode. Users are not able to enter gizmo/multi-select mode when they are in delete mode.

Impact: Ensures a clearer user interface by disabling irrelevant controls, focusing user interaction on deletion tasks.

```
1145    // Puts us into delete mode (delete key)
1146    public void SetDeleteMode(bool newDeleteMode)
1147    {
1148        deleteMode = newDeleteMode;
1149        deletionIcon.SetActive(deleteMode);
1150        GizmoSystem.instance.DisableAllGizmos();
1151        GizmoSystem.instance.DisableAllIcons();
1152        GizmoSystem.instance._modeName = GizmoSystem.ToggleMode.Default;
1153        if (deleteMode
1154            && DragSelectionManager.Instance.currentState
1155            != DragSelectionManager.DragSelectionInteractionState.AwaitingDeletionConfirmation)
1156        {
1157            DragSelectionManager.Instance.currentState = DragSelectionManager.DragSelectionInteractionState.None;
1158            DragSelectionManager.Instance.dragSelectionIcon.SetActive(false);
1159        }
1160        ObjectPlacementCancelled();
1161    }
```

6. Selection Check Update

Method Name: CheckObjectSelection()

Description: Updated condition checks to include gizmo states, preventing object selection actions when interacting with or hovering over gizmos.
Impact: Clarifies the distinction between object manipulation and gizmo interaction, enhancing the usability of the Runtime Gizmo.

```
1231    // Checks if we are clicking on the hovered object, and initializes selection with certain rules depending on object type
        Frequently called  1 usage
1232    public void CheckObjectSelection()
1233    {
1234        if (Input.GetMouseButtonDown(0) && !EventSystem.current.IsPointerOverGameObject() && GizmoSystem.instance._gizmoEnabled == false)
1235            if (Input.GetMouseButtonDown(0) && !EventSystem.current.IsPointerOverGameObject() && !RTGizmosEngine.Get.IsAnyGizmoHovered)
1236            {
```

7. Object Duplication Logic

Modifications:

Adjusted ObjectSelected() to facilitate proper multi-object duplication.

Created TriggerObjectPlacementWithoutClick() for multi-object duplication via iteration.

Impact: Streamlines the duplication process, especially for multiple objects, aligning with user expectations and improving efficiency in scene editing.

```
1447        // Runs when an object is selected. Initializes rules such as deletion and duplication, handles deletion and selection for movement
            Frequently called  12 usages
1448        public void ObjectSelected(GameObject obj, bool allowDeletion = true, bool allowDuplication = true, bool enablePimByDefault = false)
1449        {
1450            currentSelectedObject = obj;
1451            //Debug.Log($"CHUNK-ObjectSelected: currentSelectedObject {obj.name}");
1452
1453            if (deleteMode)
1454            {
1455                if (!allowDeletion) return;
1456                UndoDestroy(obj);
1457                return;
1458            }
1459
1460            inspectableObjectUIController.SetSelectedObject(obj);
1461
1462            //if (Input.GetKey(KeyCode.LeftAlt))
1463            if ((Input.GetKey(KeyCode.LeftAlt)
1464                 && DragSelectionManager.Instance.selectedObjects.Count < 2)
1465                || (Input.GetKey(KeyCode.LeftAlt)
1466                    && DragSelectionManager.Instance.currentState == DragSelectionManager.DragSelectionInteractionState.DuplicatingMultipleObjects))
1467            {
1468                //Debug.Log($"CHUNK-ObjectSelected: Object recieved {obj.name}");
1469                if (!allowDuplication)
```

8. Checking for Gizmo

A few Gizmo Related condition checks have been added in PlaceCurrentObject(), so that the objects aren't sticky and we can use the Gizmo.

```
1960                    // Placement rotation returns true if we are rotating
1961                    if (!PlacementRotation()
1962                        && GizmoSystem.instance._gizmoEnabled == false)
1963                    {
1964                        PlacementMovement();
1965                    }
1966
1967                }
1968
1969                if (Input.GetMouseButtonDown(0) && !EventSystem.current.IsPointerOverGameObject()
1970                                                && !_rotatingObject
1971                                                && GizmoSystem.instance._gizmoEnabled == false)
```

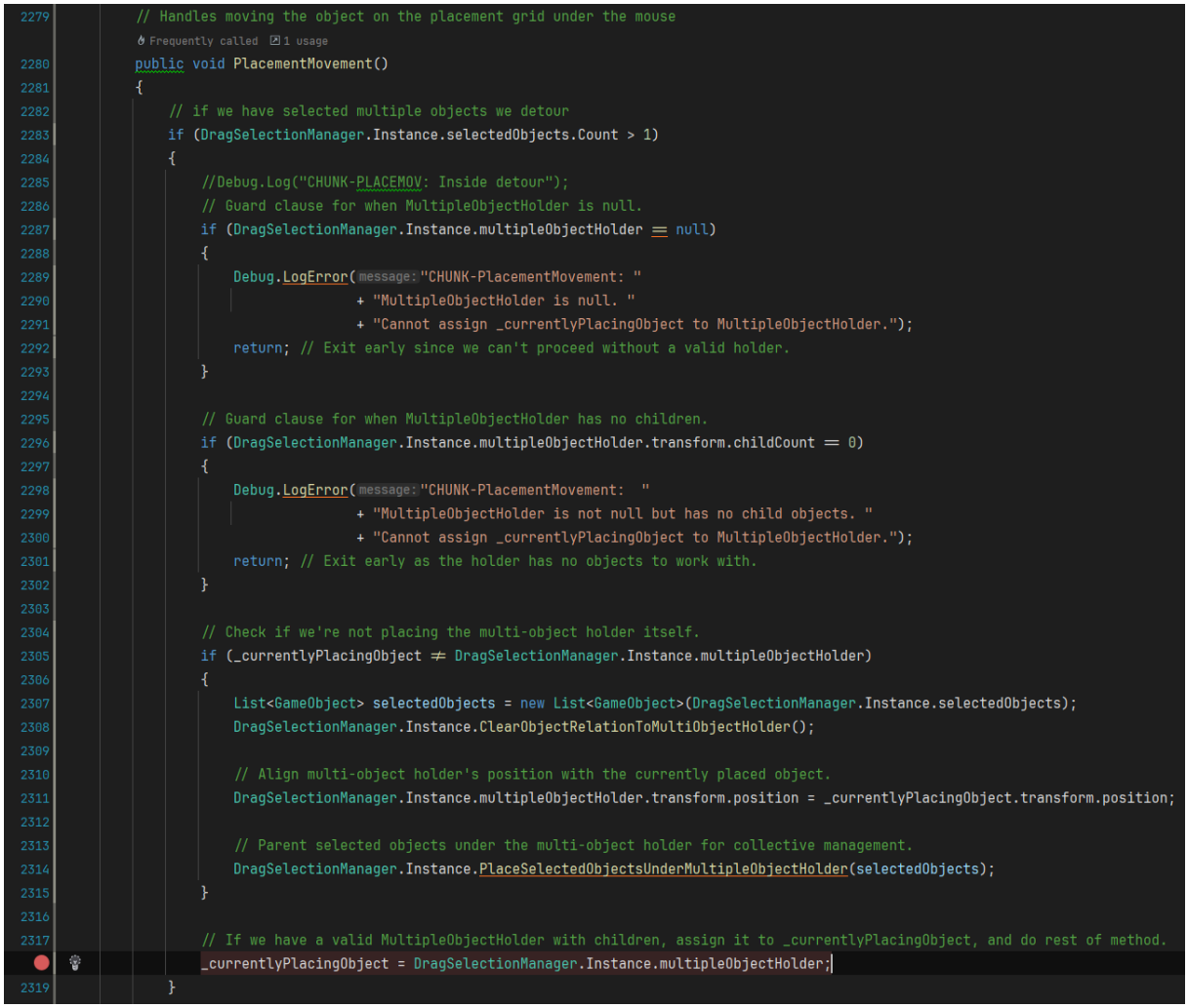
9. Placement Enhancements

Modifications:

Added detours in PlacementMovement() for multi-object handling.

Utilized multiObjectHolder for collective movement in sticky mode.

Impact: Provides a more intuitive and efficient mechanism for positioning multiple objects simultaneously, leveraging drag selection for enhanced scene manipulation.



```
// Handles moving the object on the placement grid under the mouse
Frequently called  1 usage
public void PlacementMovement()
{
    // if we have selected multiple objects we detour
    if (DragSelectionManager.Instance.selectedObjects.Count > 1)
    {
        //Debug.Log("CHUNK-PLACEMOV: Inside detour");
        // Guard clause for when MultipleObjectHolder is null.
        if (DragSelectionManager.Instance.multipleObjectHolder == null)
        {
            Debug.LogError(message:"CHUNK-PlacementMovement: "
                           + "MultipleObjectHolder is null. "
                           + "Cannot assign _currentlyPlacingObject to MultipleObjectHolder.");
            return; // Exit early since we can't proceed without a valid holder.
        }

        // Guard clause for when MultipleObjectHolder has no children.
        if (DragSelectionManager.Instance.multipleObjectHolder.transform.childCount == 0)
        {
            Debug.LogError(message:"CHUNK-PlacementMovement:  "
                           + "MultipleObjectHolder is not null but has no child objects. "
                           + "Cannot assign _currentlyPlacingObject to MultipleObjectHolder.");
            return; // Exit early as the holder has no objects to work with.
        }

        // Check if we're not placing the multi-object holder itself.
        if (_currentlyPlacingObject != DragSelectionManager.Instance.multipleObjectHolder)
        {
            List<GameObject> selectedObjects = new List<GameObject>(DragSelectionManager.Instance.selectedObjects);
            DragSelectionManager.Instance.ClearObjectRelationToMultiObjectHolder();

            // Align multi-object holder's position with the currently placed object.
            DragSelectionManager.Instance.multipleObjectHolder.transform.position = _currentlyPlacingObject.transform.position;

            // Parent selected objects under the multi-object holder for collective management.
            DragSelectionManager.Instance.PlaceSelectedObjectsUnderMultipleObjectHolder(selectedObjects);
        }

        // If we have a valid MultipleObjectHolder with children, assign it to _currentlyPlacingObject, and do rest of method.
        _currentlyPlacingObject = DragSelectionManager.Instance.multipleObjectHolder;
    }
```

New Classes Overview

GlobalUIController

Class Function: Manages global UI interactions, including registering UI controllers and minimizing UI windows.

Methods Overview

- RegisterUIController: Registers a UI controller to manage its visibility.
- MinimizeAllUIWindows: Minimizes all UI windows by toggling controller states and deactivating UI elements in the list.

Code Path Descriptions

- Minimizing All UI Windows
- Functionality Name: Minimize All UI Windows
- Trigger: User action (clicking the minimize all button)
- Flow: Awake -> RegisterUIController -> MinimizeAllUIWindows
- Outcome: All registered UI windows are minimized, and specific UI elements are deactivated.

ToggleUIButtonController

Class Function: Controls the toggling of a UI button, managing its visibility and appearance.

Methods Overview

- Awake: Initializes button image and UI element position based on startHidden.
- ToggleWindowPosition: Toggles the window's position and updates the visibility state.
- ToggleHorizontalButtonImageFlip: Flips the button image's orientation horizontally.

Code Path Descriptions

- Toggling UI Window Position
- Functionality Name: Toggle Window Position
- Trigger: User action (clicking the toggle button)
- Flow: Awake -> ToggleWindowPosition -> ToggleHorizontalButtonImageFlip
- Outcome: The UI window is shown or hidden, and the button image orientation is flipped.

ToggleUIButtonSubscriber

Class Function: Responds to toggle events from a designated ToggleUIButtonController, adjusting its position accordingly.

Methods Overview

- AdjustPositionBasedOnButtonToggle: Updates the subscriber's position in response to a toggle event.

Code Path Descriptions

- Adjusting Position Based on Button Toggle
- Functionality Name: Adjust Position Based on Button Toggle
- Trigger: Toggle event from the observed ToggleUIButtonController
- Flow: Start -> OnEnable -> AdjustPositionBasedOnButtonToggle -> OnDisable
- Outcome: The subscriber's position is adjusted according to the toggle state of the UI element.

DragSelectionManager

Class Function: Manages drag selection of objects within a scene, supporting multi-object selection, deletion, duplication, and the use of gizmos.

Methods Overview

- EnsureMultipleObjectHolder
    - Description: Ensures a holder GameObject for multiple selected objects is available and properly initialized for grouped manipulation.
- HandleMultipleDeletion
    - Description: Facilitates the deletion of multiple objects selected via drag selection, maintaining scene integrity.
- HandleMultipleDuplication
    - Description: Manages the duplication process for multiple selected objects, streamlining scene editing workflows.
- FinalizeMultiSelection
    - Description: Concludes the drag selection process, finalizing the selection of multiple objects for subsequent operations.
- ClearObjectRelationToMultiObjectHolder
    - Description: Clears the parent-child relationship between the multi-object holder and its child objects, resetting scene hierarchy.
- PlaceSelectedObjectsUnderMultipleObjectHolder
    - Description: Groups selected objects under a common parent, facilitating collective manipulation in the scene.
- FilterObjectsInSelectionRect
    - Description: Identifies and returns a list of GameObjects within the drag selection rectangle, filtering based on specific criteria.
- GetViewportBounds
    - Description: Calculates the viewport bounds based on the drag selection area, aiding in object filtering.
- IsTypingInInputField
    - Description: Checks if the user is currently typing in a TMP input field to prevent unintended state transitions.
- DidMouseClickOnUIObject
    - Description: Determines if a UI object was clicked during drag selection, ensuring UI interactions are appropriately handled.

Code Path Descriptions

- Drag Selecting Objects
    - Functionality Name: Drag Selecting
    - Trigger: Press and hold the left mouse button (without pressing Ctrl or interacting with a gizmo).
    - Flow: Update -> HandleStateActions -> HandleDragSelection
    - Outcome: Objects within the drag selection area are identified and can be manipulated as a group.
- Deleting Multiple Objects
    - Functionality Name: Deleting Multiple Objects
    - Trigger: Pressing the Delete key with multiple objects selected.
    - Flow: HandleStateTransitionsFromKeyPress -> HandleDeletionConfirmation -> HandleMultipleDeletion
    - Outcome: Selected objects are deleted after confirmation.
- Duplicating Multiple Objects
    - Functionality Name: Duplicating Multiple Objects
    - Trigger: Pressing the LeftAlt key with multiple objects selected.
    - Flow: HandleStateTransitionsFromKeyPress -> HandleDuplicationConfirmation -> HandleMultipleDuplication
    - Outcome: Creates duplicates of the selected objects.

GizmoSystem

Class Function: Manages gizmos for object manipulation (move, rotate, scale) in a Unity scene, enabling dynamic interaction based on user input and selection.

Methods Overview

- HandleGizmoFromKeyPress
    - Description: Handles user input to enable different gizmo modes and manage gizmo behavior.
- SelectAndEnableGizmo
    - Description: Enables gizmo only when the user selects an object.
- SwitchGizmoModes
    - Description: Toggles between different gizmo modes (move, rotate, scale) based on user input, adapting to editing needs.
- SwitchIcon
    - Description: Switches the visibility of icons based on the specified toggle mode.
- HoverAndSelectAnObject
    - Description: Retrieves the object under the mouse cursor, facilitating its selection or manipulation with an active gizmo.
- SelectObjects
    - Description: Manages the selection of objects for manipulation with the currently active gizmo, checking if scale is needed, refining user interaction.
- Selecting
    - Description: Handles the selecting of objects and updates the selection state. Handles the gizmo ctrl multi-select/deselect.
- OnSelectionChanged
    - Description: Updates the state and target objects of the active gizmo following a change in the selection, ensuring correct gizmo behavior.
- SetWorkGizmoId
    - Description: Sets the currently active gizmo based on the selected mode, aligning manipulation capabilities with user intent.
- CtrlKeyForDragSelect
    - Description: Handles the addition or removal of objects to the selection via Ctrl-click in multi-gizmo mode, enhancing selection flexibility.
- IsTypingInInputField
    - Description: Checks if the user is currently typing in a TMPro input field.
- CheckScale
    - Description: Checks if the picked object is of type "ObstacleDesignerShape".

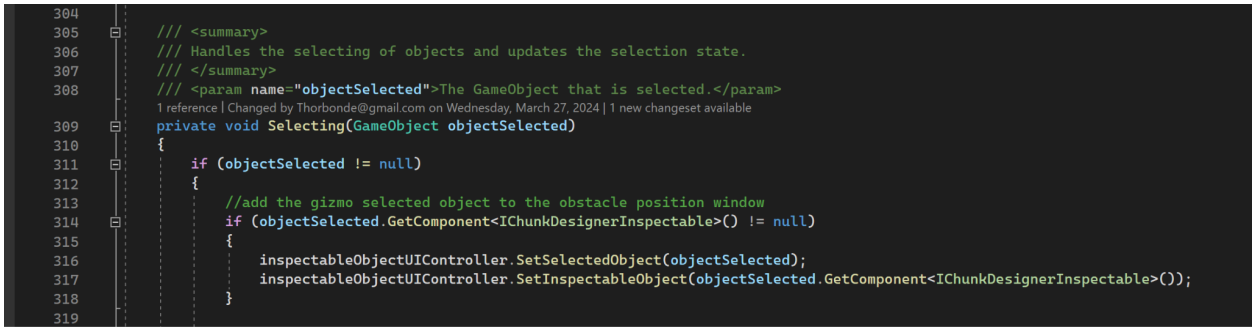
- CheckAmountOfODShapeInList
    - Description: Checks the number of ObstacleDesignerShape objects in the list of selected objects.
- ClearSelection
    - Description: Clears the current selection, disables gizmos and icons, and resets mode state.
- DisableAllGizmos
    - Description: Deactivates all gizmos, resetting the manipulation state to avoid interference during non-manipulation actions.
- DisableAllIcons
    - Description: Hides all gizmo-related icons, streamlining the UI to reflect the current state of gizmo activation.

Code Path Descriptions

- Enabling Gizmo
    - Functionality Name: Enabling Move Gizmo
    - Trigger: Pressing the E/R/T keys
    - Flow: Update -> HandleGizmoFromKeyPress -> SwitchGizmoModes -> SelectAndEnableGizmo -> SetWorkGizmoId -> SelectObjects -> Selecting -> OnSelectionChanged
    - Outcome: The move/rotation/scale gizmo is enabled, allowing the user to transform selected objects in the scene.
- Object Selection and Gizmo Activation
    - Functionality Name: Object Selection and Gizmo Activation
    - Trigger: Left mouse click on an object
    - Flow: Update -> HandleGizmoFromKeyPress -> SelectAndEnableGizmo -> SelectObjects -> Selecting -> OnSelectionChanged
    - Outcome: Selected objects are manipulated using the active gizmo mode (move, rotate, scale).

Uses of InspectableObjectUIController inside GizmoSystem

We also updated the selected object position/rotation window so that when the user selects an object in any gizmo mode, the object's information will show up in the window.



```
304
305     /// <summary>
306     /// Handles the selecting of objects and updates the selection state.
307     /// </summary>
308     /// <param name="objectSelected">The GameObject that is selected.</param>
        1 reference | Changed by Thorbonde@gmail.com on Wednesday, March 27, 2024 | 1 new changeset available
309     private void Selecting(GameObject objectSelected)
310     {
311         if (objectSelected != null)
312         {
313             //add the gizmo selected object to the obstacle position window
314             if (objectSelected.GetComponent<IChunkDesignerInspectable>() != null)
315             {
316                 inspectableObjectUIController.SetSelectedObject(objectSelected);
317                 inspectableObjectUIController.SetInspectableObject(objectSelected.GetComponent<IChunkDesignerInspectable>());
318             }
319
```

The GizmoSystem and DragSelectionMangaer scripts with the icon images are in a folder we created with the path:

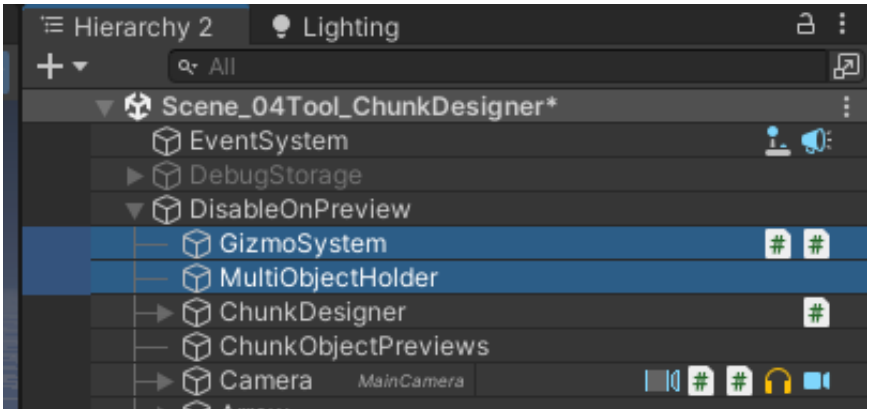
Assets/ProjectFiles/_05Scripts/LevelDesigner/ChunkDesigner_Scene/Gizmo

Changes to Hierarchy

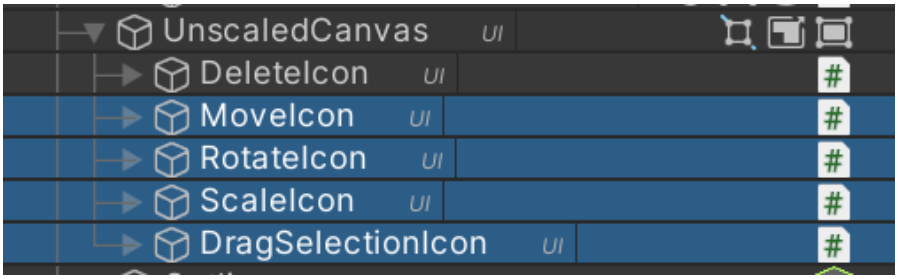
A few changes have been made to the hierarchy:

The object GizmoSystem has been added and has the GizmoSystem script and the DragSelectionManager script attached.

The MultiObjectHolder is utilized for the manipulation of multiple objects.



Four icon objects are added under the UnscaledCanvas with the MouseIcon script attached.

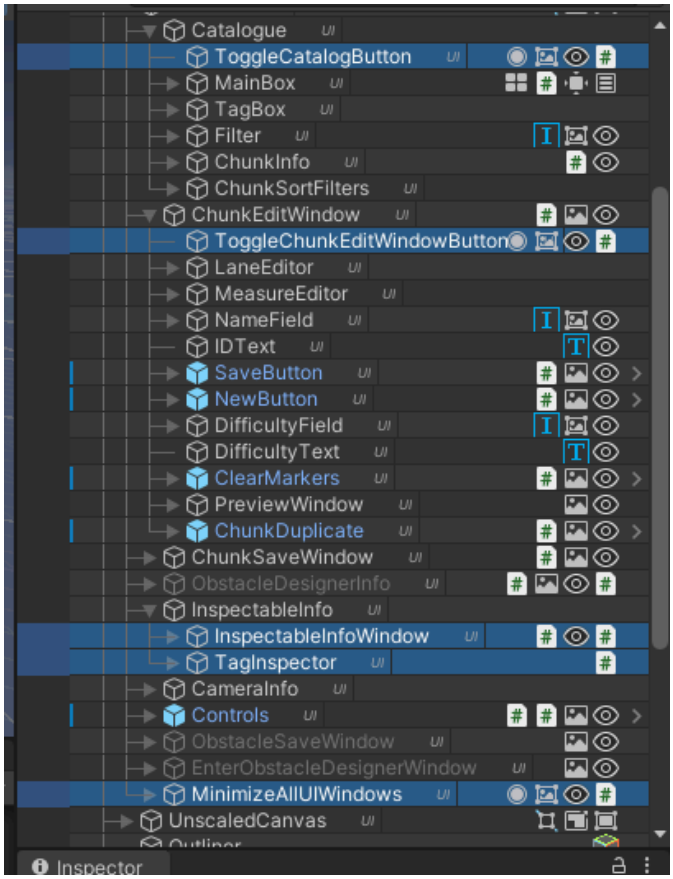


UI Minimization

In the Catalog and the ChunkEditWindow, we have added a button equipped with the ToggleUIButtonController script.

At the bottom of the canvas itself we have added a "Minimize All" button, which has the GlobalUIController script.

The InspectableInfoWindow and TagInspector are equipped with the ToggleUIButtonSubscriber script, ensuring they will synchronize with the ChunkEditWindow.

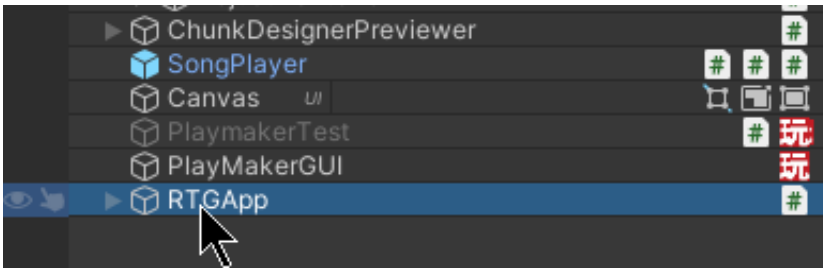


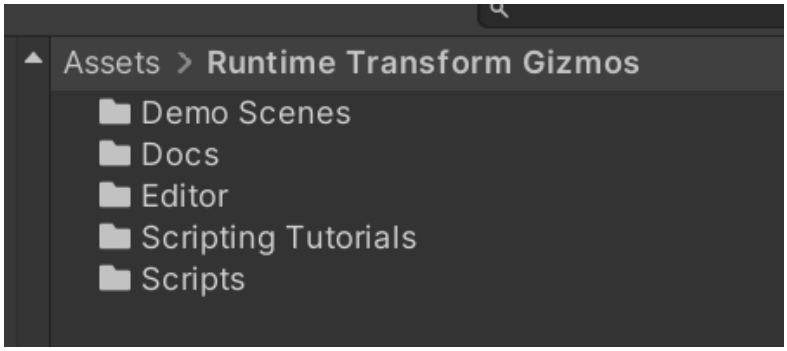
Runtime Transform Gizmo Plugin

Overview

The Runtime Transform Gizmo plugin was bought in the Unity Asset Store. It contains a lot of great features that can manipulate objects at runtime.

RTGApp has been introduced as the object that holds all the management scripts for the Runtime Gizmo Application.

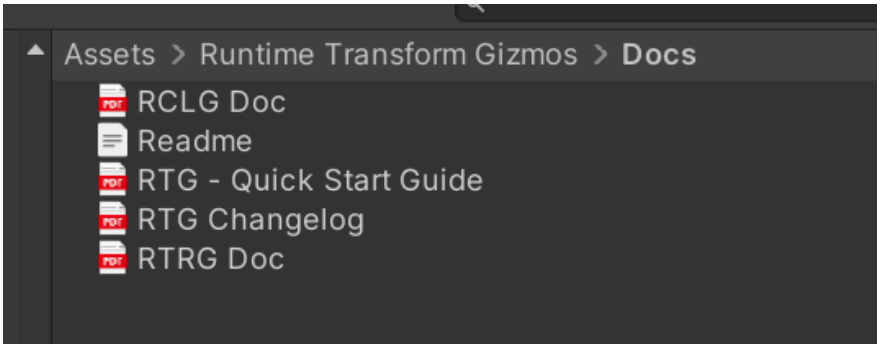




The *Scripting Tutorials* folder contains all his scripts to use the plugin from his tutorials.

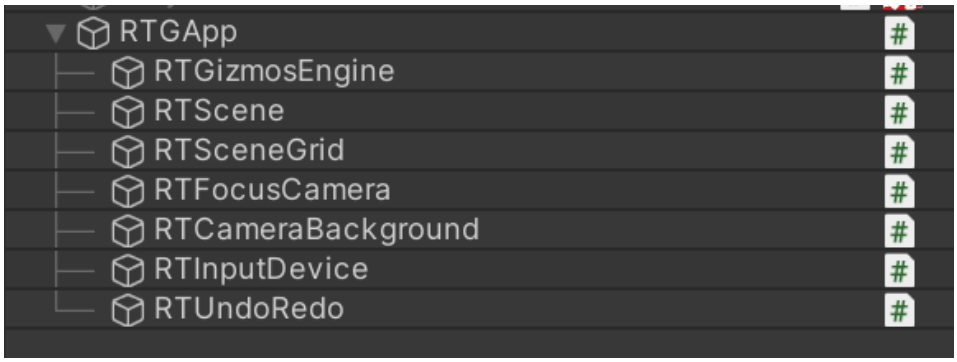
The *Scripts* and *Editor* folder contains all the plugin's code.

Inside the *Assets/Runtime Transform Gizmos/Docs* folder, there are some detailed instructions about the plugin's code.



RTGApp in the Hierarchy

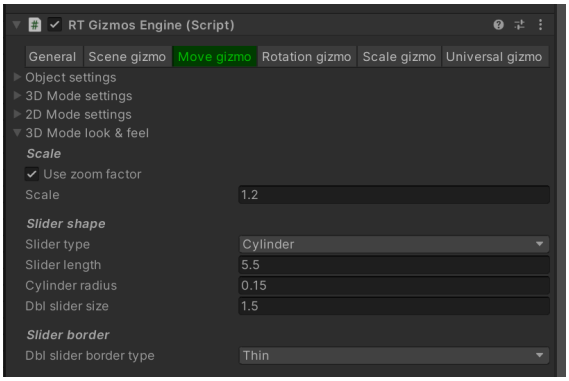
There are seven GameObjects inside the RTGApp:

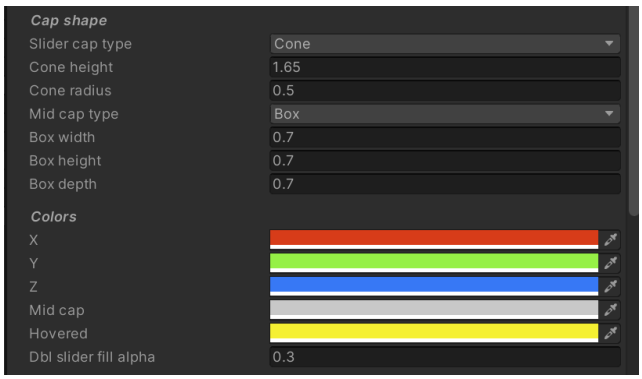


RTGizmosEngine

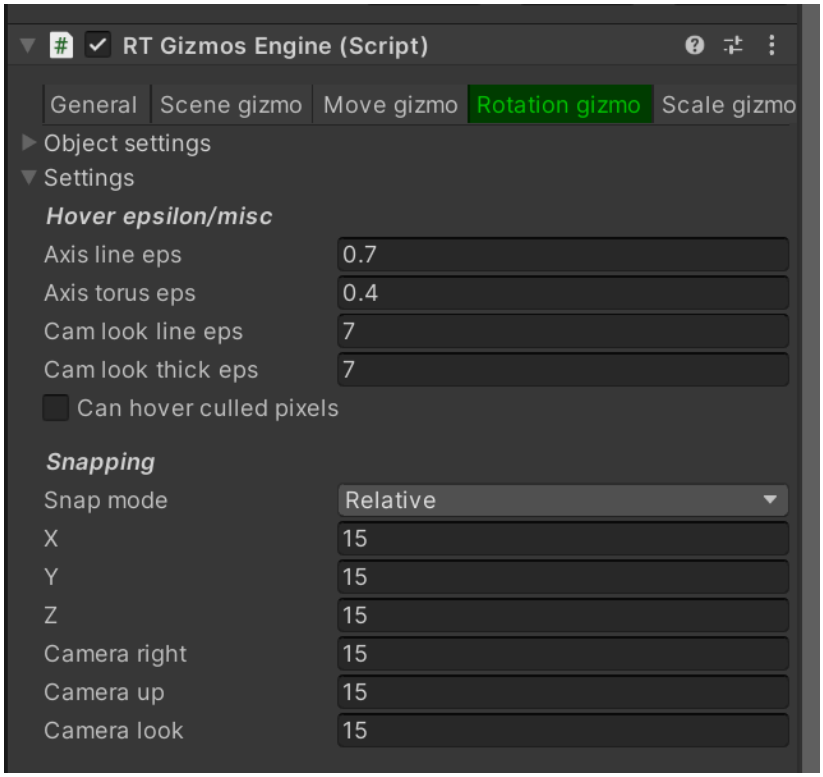
The RTGizmosEngine contains all the settings for the gizmo.

Inside the Move gizmo/3D Mode look & feel, you can change the size, colors, radius, shape, etc. of the move gizmo. The same applies for the rotate and scale gizmo.

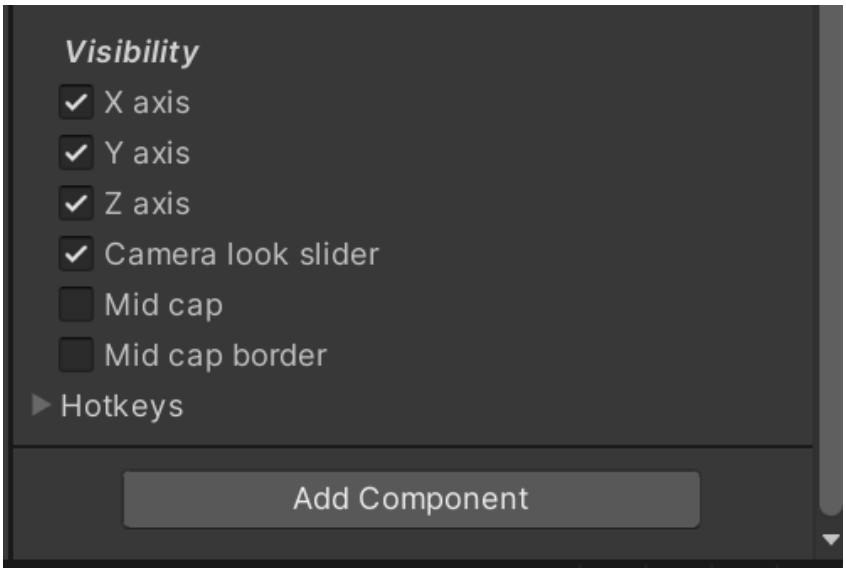




The rotation snapping angle mentioned in the user guide can be changed in the Rotation gizmo section inside the RTGizmosEngine. Currently, the snapping angle is set to 15 degrees for all the directions.

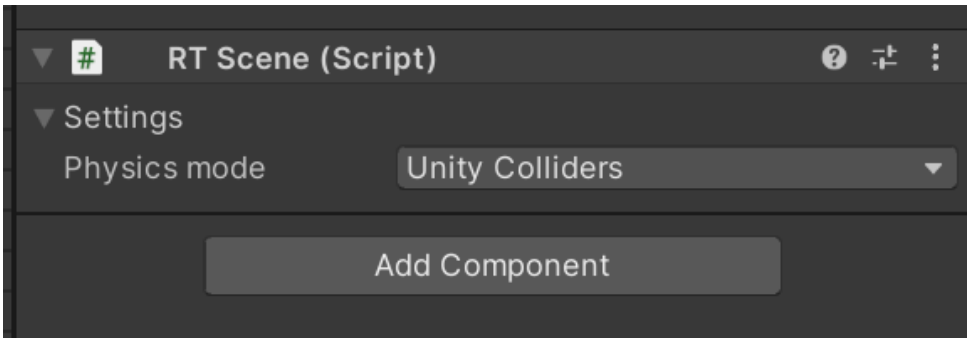


The visibility of parts of the rotation gizmo can be changed in the Visibility section. We turned off the mid cap and mid cap border, which is the ability to rotate objects when users drag on the positions inside the circles.



RTScene

We changed the *Physics mode* in RTScene to Unity Colliders to solve the preview bug (things moving slowly).



RTSceneGrid

The RTSceneGrid has the settings for the scene grid. You can change the size, color and hotkeys of it.

RTInputDevice

This contains all the inputs from the plugin through the code path:
RTInputDevice/MouseInputDevice/RTInput

RTUndoRedo

The RTUndoRedo GameObject handles the undo system in the plugin. However, we disabled it in the *RTGApp* script because of the conflict with the editor's undo system.

Testing and Results

We have conducted many tests with both our internal level designers and Buffalo Buffalo's level designers. Every new feature undergoes internal testing by our team, primarily through integrating the new code into their branch once we consider it stable enough for an internal release. The level designer then employs the new function while designing a level, deliberately attempting to break it with the intent of uncovering any minor flaws. Once deemed stable, it is showcased to the client during meetings, where they can test, provide feedback, and suggest adjustments. Additionally, minor improvements are continuously tested throughout the week by uploading them to an external testing branch, with the client's level designer being informed of any adjustments that need to be tested. The ultimate test of our enhancements to the level designer tool is conducted by Buffalo's newly hired level designer who has only had experience with the unimproved tool.

Interviews will usually be formed after testing and feedback will be given.

Added Controls

- Q: Enter/Exit Box Drag to select multiple objects mode.
- E, R, T: Enter/Exit Move, Rotate, Scale Gizmo modes.
    - Scale only applicable in obstacle designer mode
- X: Hard exit all modes, resetting back to default.
- ALT: Enables duplication of multiple selected objects.
- DELETE: Deletes multiple selected objects.

If Mirror is added:

- M: Mirrors the position of multiple objects.
- CTRL+M: Mirrors the position of multiple objects using their common center line as mirror line.

User Guide

Note that these are just the basic instructions on how to use the new features.

Gizmo

The move, rotate, scale gizmo features are activated by pressing the hotkeys ERT. Icons will be shown after you press the hotkeys, indicating the mode you are in. However, to activate the actual gizmo, you need to select an object that you want to transform first. Also note that scale gizmo is designed to be only used on objects that have the “obstacle designer shape” type.

To Activate & Use Gizmo

1. Enter gizmo mode directly

   Press E to enter Move gizmo mode,

   Select an object

   A yellow highlight will be shown when you ***hover*** over the gizmo

   Drag the arrows in three dimensions to move objects

   Drag the squares at the bottom of the gizmo to move the objects in two dimensions at a time

   or

   Press R to enter Rotation gizmo mode,

   Select an object

   A yellow highlight will be shown when you ***hover*** over the gizmo

   Drag the circles in three dimensions to rotate objects

   Hold Left Ctrl and ***drag*** the gizmo to snap your rotation by 15° (the angle can be adjust easily in the hierarchy, see RTGAPP Section for more details)

   or

   Inside obstacle designer, ***Press T*** to enter Scale gizmo mode,

   Select an object (you can only use scale gizmo for objects inside the obstacle designer category; if you click on other objects, it will disable gizmo)

A yellow highlight will be shown when you ***hover*** over the gizmo

Drag the arrows in three dimensions to scale objects

Drag the cube in the middle of the gizmo to scale objects in three dimensions at a time

2. In gizmo mode, ***Toggle*** between different modes by toggling between the different keys ERT

   *Note that you can only toggle to scale gizmo inside obstacle designer **AND** the object selected is an object inside the obstacle designer category.*

To Exit Gizmo

1. Exit gizmo mode with the same keys

   Press E to exit Move gizmo mode

   Press R to exit Rotation gizmo mode

   Inside obstacle designer, ***Press T*** to exit Scale gizmo mode

2. Exit gizmo with the universal X

   Press X to exit all the modes, including move, rotate, scale gizmo.

3. Exit gizmo by clicking on the environment or the catalog

   Clicking on the screen or UI will exit gizmo as well.

Ctrl Multi-select in Gizmo

Inside any gizmo modes, you can use the left control key to select/deselect multiple objects

Hold Left Ctrl and ***Click*** on an object to select/deselect

Multi-selection drag box

The drag-selection box is activated by the hotkey Q. When the user presses Q, an icon will be shown to indicate they are in drag-select mode. After the user selects objects, they enter multiple objects selected mode, where the user can do a series of multi functions described below.

To Activate & Use drag-selection box

Press Q to enter drag-selection box mode

Drag and ***select*** objects

The selected objects will be highlighted

Click on any object selected to sticky move all the objects selected together

To Exit Drag-Selection Box or Multiple Objects Selected Mode

1. Exit gizmo mode with the same key

   Press Q to exit both modes.

2. Exit gizmo with the universal X

   Press X to exit all the modes.

3. Exit both modes by clicking on the catalog

   Clicking on UI will exit gizmo

   Note that you cannot exit multi-select mode by clicking on screen

Multi-gizmo

Multi-gizmo is a combined feature that allows the user to use drag-selection box, ctrl-select/deselect and gizmo together.

To Activate multi-gizmo

1. Enter gizmo mode directly

   In multiple objects selected mode,

   Press E to enable multi-Move gizmo

   The move gizmo will show up directly, and you can still use all the functionalities in single gizmo mode.

   or

   Press R to enable multi-Rotation gizmo

   The rotation gizmo will show up directly, and you can still use all the functionalities in single gizmo mode.

   or

Inside obstacle designer, ***Press T*** to enable multi-Scale

The scale gizmo will show up directly, and you can still use all the functionalities in single gizmo mode.

In multi-gizmo mode, ***Toggle*** between different modes by toggling between the different keys ERT

*Note that you can only toggle to scale gizmo inside obstacle designer **AND** the object selected is an object inside the obstacle designer category.*

To Exit Gizmo mode inside multi-gizmo

1. Exit gizmo mode with the same keys

   Press E to exit Move gizmo mode

   Press R to exit Rotation gizmo mode

   Inside obstacle designer, ***Press T*** to exit Scale gizmo mode

To Exit multi-gizmo mode

1. Exit multi-gizmo with the same key

   Press Q to exit multi-gizmo.

2. Exit multi-gizmo with the universal X

   Press X to exit all the modes, including all the gizmo modes and multiple objects selected mode.

   Exit multi-gizmo by clicking on the catalog

   Clicking on the UI will exit multi-gizmo as well.

   Note that you cannot exit multi-gizmo mode by clicking on screen

Ctrl Multi-select in multi-Gizmo

Inside multi-gizmo, you can use the left control key to select/deselect multiple objects.

Inside multi-gizmo mode,

Hold Left Ctrl and ***Click*** on an object to select/deselect

Multi-deletion

Multi-deletion is a feature inside the multiple objects selected mode. You can delete all the multiple objects at the same time by pressing delete inside the multi-mode.

> ***Press Delete*** inside multi-mode.
>
> The multiple objects selected will have a red highlight indicating the objects that you selected and want to delete.
>
> ***Click*** on one of the objects selected to delete all.

Multi-duplication

Multi-duplication is a feature inside the multiple objects selected mode. You can duplicate multiple objects by pressing alt and clicking on one of the objects.

> ***Press Alt*** and ***click*** on the multiple selected objects.
>
> (The duplicated objects will be in the original objects position and the original objects will move with the mouse. This will be discussed in the known issue)

Multi-mirror mode

Multi-mirror mode is a feature newly implemented inside the multiple objects selected mode. It still needs some testing so we put it inside a side branch (changeset 3627). You can mirror the objects' position by pressing M.

> ***Press M*** when having multiple objects selected. This will move them to the mirrored position using The Center Lane as the mirror.
> Obstacles will have their internal scale flipped.
>
> ***Press M +*** **(left or Right)*Ctrl*** when having multiple objects selected.This will move them to the mirrored position using the line going through the multiple selected object average center as the mirror, therefore making them rotate around their own position.
> Obstacles will have their internal scale flipped.

UI minimize windows

To minimize the windows

1. ***Click*** on the arrow buttons on the side of the windows.
   The left arrow button controls all the windows on the left.

   The right arrow button will minimize the ChunkID window and the Preview/Listen window. The obstacle designer window and the position/rotation/scale window will move together to the right when you minimize the right windows.

2. The minimize all windows button on the top right corner will minimize all the windows except the obstacle designer window.

To re-open the windows

Click on the arrow buttons on the side of the windows.

Known Bugs, Issues and Workaround

A List of any known bugs or issues that were not fixed and/or provided any temporary workarounds.

1. Enemy Rotation direction

When utilizing the rotation gizmo to rotate enemies in the x and z directions, the enemy may rotate in the opposite direction from the intended one. For instance, when dragging the gizmo forwards, the enemy rotates backward. However, upon entering preview mode, the rotation remains consistent regardless of the manipulation performed in the editor. Unlike other object types in the editor which possess modifiable position, rotation, and scale values, each enemy has its own predefined set of animations and spawn methods. Consequently, these enemies are not affected by changes in rotation. A possible workaround would involve disabling the rotation gizmo for enemy objects.

2. Pimmarker Not Rotating with Objects

When rotating objects in single-object rotation or multi-object rotation, the markers (Pimmarkers, Timmarkers, etc.) do not rotate with the objects. This is because when we are finding the mesh center to set as the pivot point for rotation gizmo, we are checking the PIMMarker - a method similar to the rotationTransform in the ChunkDesigner_Scene.

```
foreach (GameObject obj in _selectedObjects)
{
    var pimMarker = obj.GetComponentInChildren<PIMMarker>();
    if (pimMarker != null)
    {
        _selectedObjectsRotationAndScale.Add(pimMarker.targetObject);
    }
    else
    {
        _selectedObjectsRotationAndScale.Add(obj);
    }
}
```

We are trying to rotate the object from each mesh center, not considering the markers. For single objects, our considerations were: we probably don't want to make the users rotate an object in a way that makes the objects' marker behind the object self. Since the obstacles usually appear when the player hits one of the markers, it will cause the

object to appear behind the player which might cause problems. The markers are used for checking when the player hits and trigger entering animation, so we won't want the player not being able to touch the markers. Therefore, we decided to keep the markers unrotated. However, rotating objects, especially multiple ones, by their total mesh center, might be confusing since it will be hard to determine how the markers and objects connect with each other. And sometimes we want the markers to move along.

This issue was only raised a week ago, so we haven't had much time to delve into it, as it seems quite complex to solve. However, some potential workarounds could involve finding the entire object during rotation (but this might cause pivot change) or isolating the markers individually and adjusting their orientation in accordance with the rotation of the objects.

Additionally, it might be beneficial to conduct testing and involve level designers in assessing the situations where rotation is most critical and determining whether it's necessary to synchronize the markers' rotation with the objects.

3. Multi Duplication mouse position

When duplicating multiple objects, if user click an object not being duplicated the duplicate objects will have the position to mouse offset by position of clicked objects

4. Multi duplication

ORIGINAL objects move with the mouse after click, not duplicated ones

5. Multi Duplication and clicking on UI

When having duplicated multiple objects if UI is clicked, user will lose stickiness of the duplicated objects, and they will be put back to where they were duplicated from, effectively hiding behind the originals

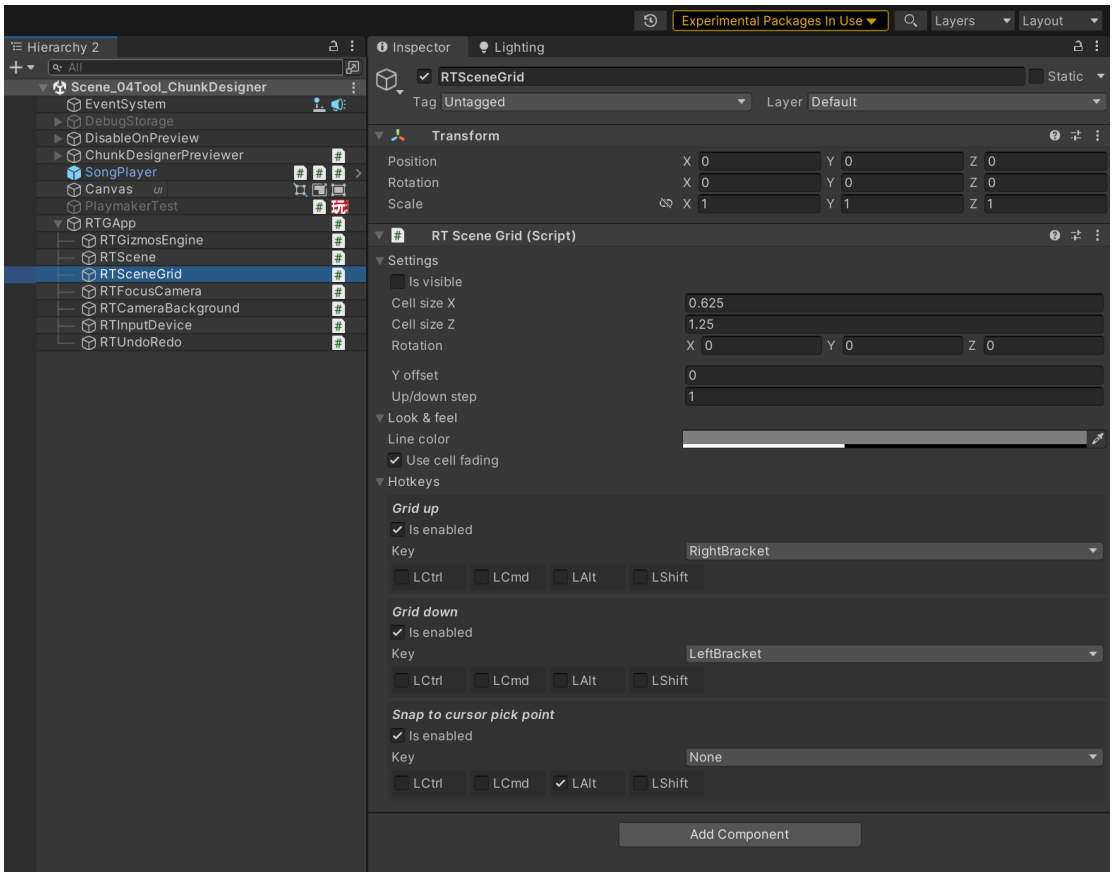
6. Snapping

The Gizmo Asset (Runtime Transform Gizmos) has a snapping function in it with a scene grid. We changed the grid to the editor’s grid size with 0.625 for the x dimension and 1.25 for z dimension. The snapping in the current version will snap to a 1 unit grid that is generated where the selected object is put.

To make the scene grid visible in the settings

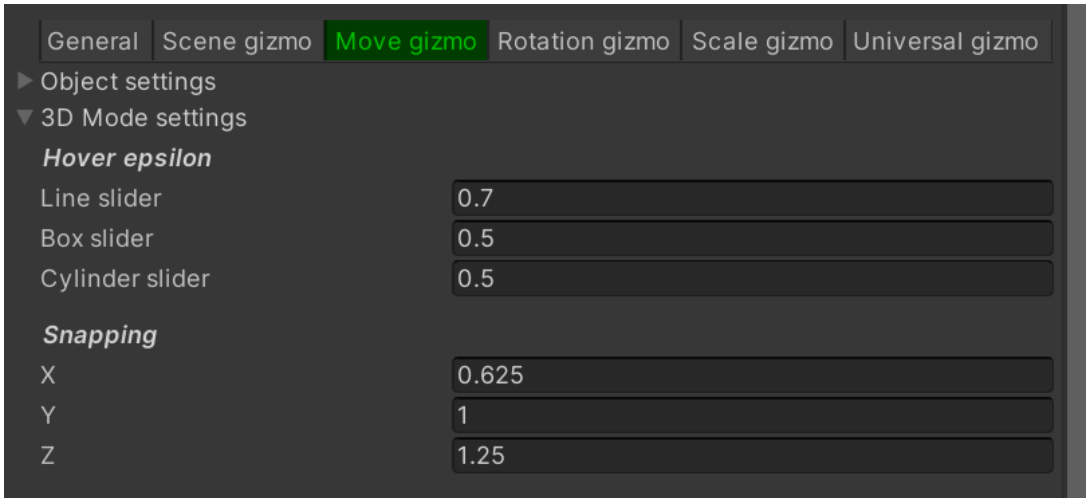
Go to RTGApp -> RTGSceneGrid, and click on “is visible”

The grid size can be changed by changing cell size x and cell size z.



To make the snapping working

Go to RTGApp -> RTGizmosEngine -> Move Gizmo, and change the X and Z inside Snapping to 0.625 and 1.25.

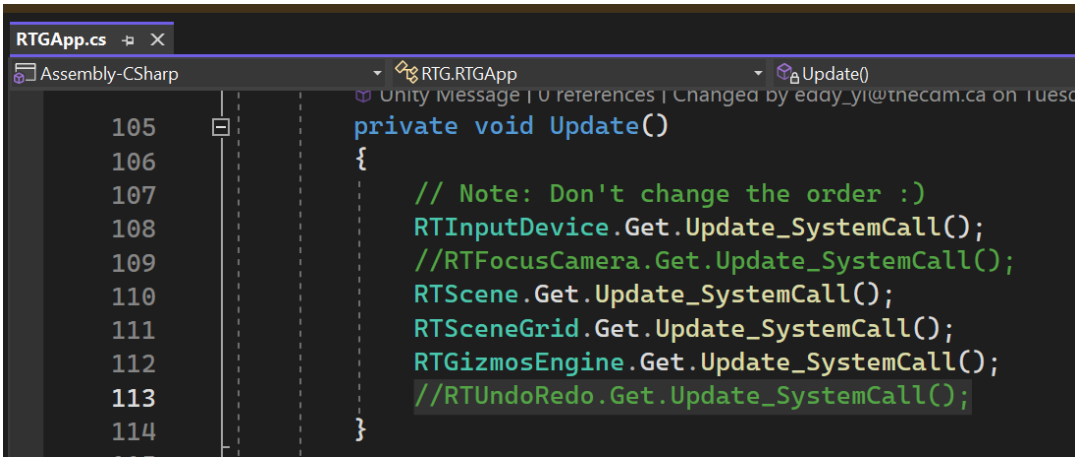


Instructions on how to use snapping in Move Gizmo

Hold down Left Ctrl in move gizmo mode, and ***drag*** the arrows.

7. Undo - Gizmo

The Gizmo Asset (Runtime Transform Gizmos) has an undo function, but it was conflicting with the editor's undo. Due to time limitations and priorities, we didn't have much time to look into it. We disabled the undo feature by commenting out the line "RTUndoRedo.Get.Update_SystemCall();" inside RTGApp.cs Update() function. It will also cause issues - the preview will become extremely slow, if we uncomment the line.



8. Undo - Multi-select

The Undo system can stop working when using undo after having been multi deleted or multi duplicated.

Future Considerations

Refactor DragSelectionManager into A DragSelectionHandler and MultiObjectManipulation class, as that would give a better separation of concerns.

TriggerObjectPlacementWithoutMouseCLick() in Chunky, should use the version that has been commented out, as much of the code is repeat code.

MirrorMode will flip unsymmetrical obstacles, by using scale, a better approach should be implemented.

Conduct testing and interviews with the level designers to see how they want the markers to be rotated.

Contact Information

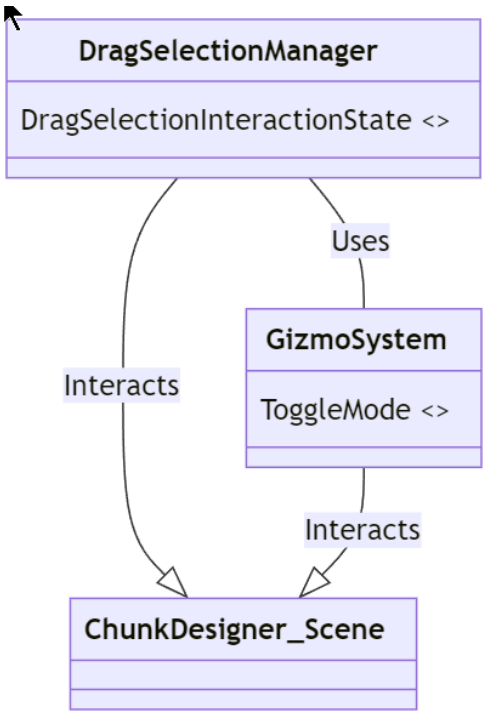
For further information or queries related to this document or the Chunk Editor enhancements, please contact:

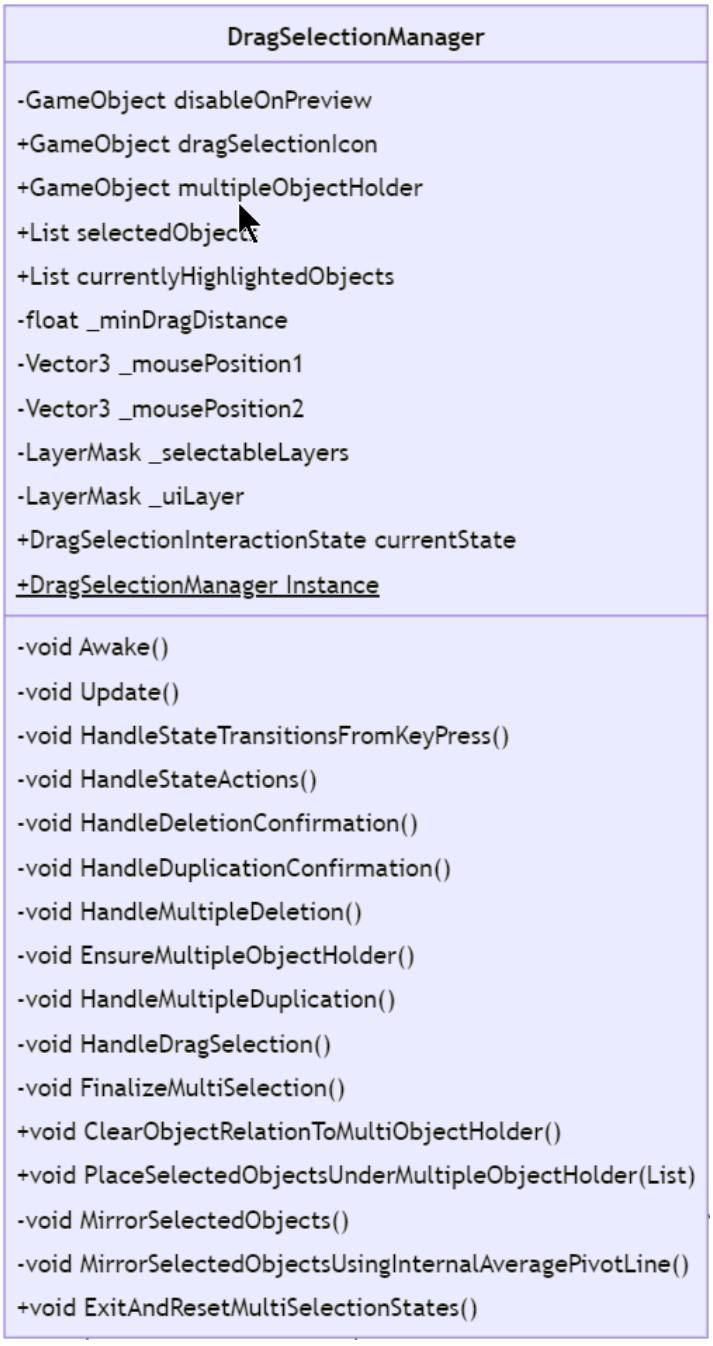
Developers: Zimo Wu & Mpho Thor Bonde

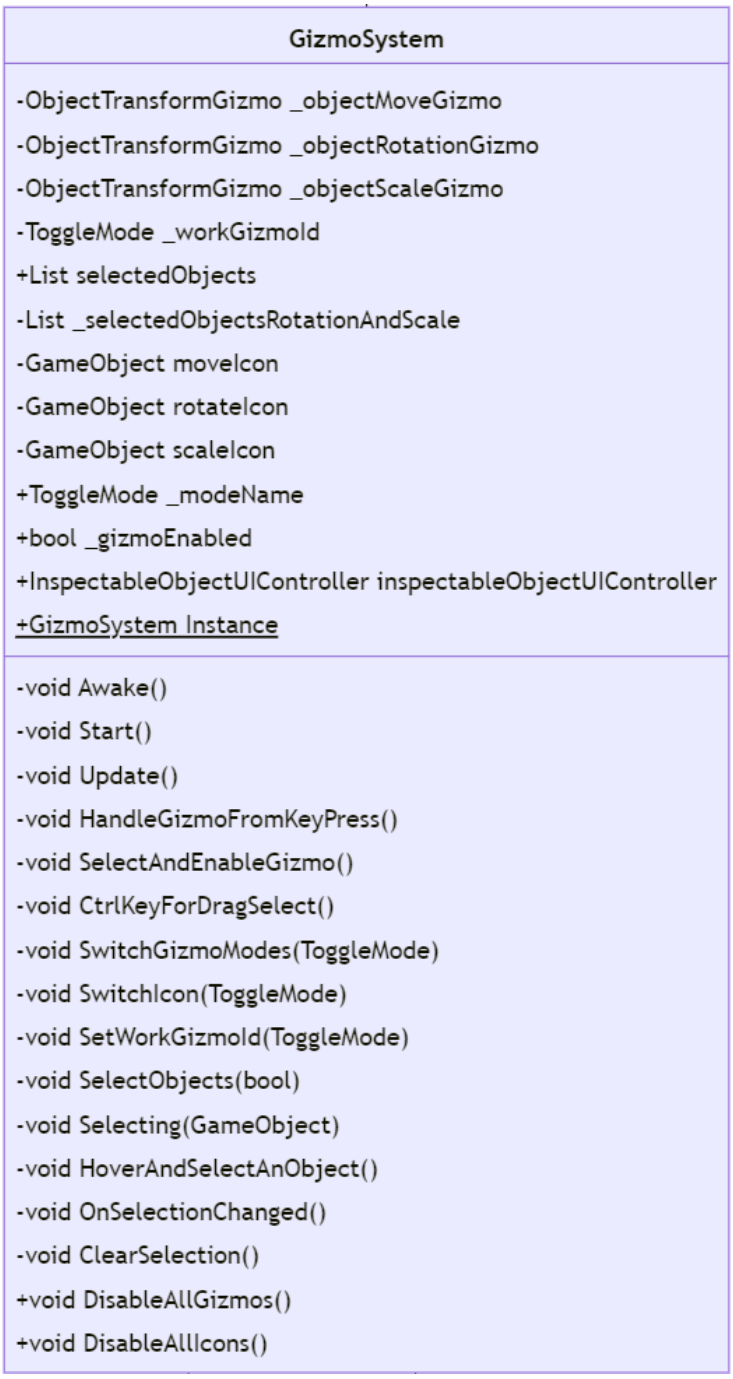
Email: zimoyuhu@gmail.com & Thorbonde@Gmail.com

Appendix

The following are the class relationships between the newly created classes.


DragSelectionManager
DragSelectionInteractionState <>
Uses
GizmoSystem
ToggleMode <>
Interacts
Interacts
ChunkDesigner_Scene


DragSelectionManager
-GameObject disableOnPreview
+GameObject dragSelectionIcon
+GameObject multipleObjectHolder
+List selectedObjects
+List currentlyHighlightedObjects
-float _minDragDistance
-Vector3 _mousePosition1
-Vector3 _mousePosition2
-LayerMask _selectableLayers
-LayerMask _uiLayer
+DragSelectionInteractionState currentState
+DragSelectionManager Instance
-void Awake()
-void Update()
-void HandleStateTransitionsFromKeyPress()
-void HandleStateActions()
-void HandleDeletionConfirmation()
-void HandleDuplicationConfirmation()
-void HandleMultipleDeletion()
-void EnsureMultipleObjectHolder()
-void HandleMultipleDuplication()
-void HandleDragSelection()
-void FinalizeMultiSelection()
+void ClearObjectRelationToMultiObjectHolder()
+void PlaceSelectedObjectsUnderMultipleObjectHolder(List)
-void MirrorSelectedObjects()
-void MirrorSelectedObjectsUsingInternalAveragePivotLine()
+void ExitAndResetMultiSelectionStates()


GizmoSystem
-ObjectTransformGizmo _objectMoveGizmo
-ObjectTransformGizmo _objectRotationGizmo
-ObjectTransformGizmo _objectScaleGizmo
-ToggleMode _workGizmoId
+List selectedObjects
-List _selectedObjectsRotationAndScale
-GameObject moveIcon
-GameObject rotateIcon
-GameObject scaleIcon
+ToggleMode _modeName
+bool _gizmoEnabled
+InspectableObjectUIController inspectableObjectUIController
+GizmoSystem Instance
-void Awake()
-void Start()
-void Update()
-void HandleGizmoFromKeyPress()
-void SelectAndEnableGizmo()
-void CtrlKeyForDragSelect()
-void SwitchGizmoModes(ToggleMode)
-void SwitchIcon(ToggleMode)
-void SetWorkGizmoId(ToggleMode)
-void SelectObjects(bool)
-void Selecting(GameObject)
-void HoverAndSelectAnObject()
-void OnSelectionChanged()
-void ClearSelection()
+void DisableAllGizmos()
+void DisableAllIcons()