Osnove virtualnih okruženja

2. Laboratorijska vježba

Opis skripte PlayerCollision.cs

Unutar funkcije Start() dodala sam PlayerPrefs.SetInt() koji za oznaku „print“ postavlja vrijednost 1. Tu vrijednost ću kasnije koristiti za provjeru je li došlo do promjene u hijerarhiji. Unutar funkcije OnCollisionEnter dodala sam GameObject child koji je parametar collision pretvoren u gameObject. Uvjet selekcije if() provjerava ima li objekt child Tag s vrijednosti „Sticky“, zatim provjerava je li masa childMass manja od katamariMass te posljednje provjerava nalazi li se u Dictionaryju objekt s ključem childName. Unutar tijela selekcije if() PlayerPrefs za oznaku „print“ se postavlja na 1, jer će sigurno doći do promjene hijerarhije jer je došlo do sudara objekata. U Dictionary se dodaje objekt s ključem childName i vrijednosti childMass. Stvara se Collider col u koji se sprema collider koji se dohvaća iz parametra collision i funkcije GetContact(0) i thisCollider. Collider col je prvi collider koji je detektirao sudar u točki. Nadalje objektu child dohvaća se komponenta transform i komponenta parent. Vrijednost komponente parent se postavlja na col.GetComponent<Transform>(). Na taj način se objektu koji se sudario s Katamari, upravo Katamari postavlja kao roditelj unutar hijerarhije. Nadalje se mijenja masa Katamari pomoću funkcije UpdateKatamariMass(childMass, „add“). Parametri koje predajemo su masa childMass, odnosno masa koju nadodajemo katamari, te string „add“ koji označava dodavanje. Posljednja naredba je Destroy(child.GetComponent<Rigidbody>()) koji uništava komponentu Rigidbody objekta child.

Skripta PlayerCollision.cs

```
using System.Collections.Generic;
using UnityEngine;

/// <summary>
/// Class that implements actions during a collision.
/// </summary>
public class PlayerCollision : MonoBehaviour
{
    [SerializeField] private GameObject _stickyItems = null;
    private Dictionary<string, float> _oldMassesOfStickyItems = null;

    /// <summary>
    /// Start is called before the first frame update.
    /// Initialize storage for masses of sticky items.
    /// </summary>
    private void Start()
    {
        _oldMassesOfStickyItems = new Dictionary<string, float>();
        PlayerPrefs.SetInt("print", 1);
    }
```

```
    /// <summary>
    /// On a collision check if the collision object has a tag "Sticky",
otherwise ignore it.--done
    /// If an object has a tag "Sticky" and has a smaller mass than
katamari ball, add it to the katamari ball,
    /// store it's mass to the storage and add it to the katamari ball's
mass.
    /// </summary>
    /// <param name="collision">Object that katamari ball has collided
with.</param>
    private void OnCollisionEnter(Collision collision)
    {
        float childMass =
collision.gameObject.GetComponent<Rigidbody>().mass;
        string childName = collision.gameObject.name;
        float katamariMass = GetKatamariMass();
        GameObject child = collision.gameObject;

        // consider the collision only if the object is sticky, its mass is
smaller than katamariMass and if the sticky object is not already stuck on
the ball
        if (child.tag.Equals("Sticky") && childMass<katamariMass &&
!_oldMassesOfStickyItems.ContainsKey(childName))
        {
            PlayerPrefs.SetInt("print", 1);
            // add the collided object to the _oldMassesOfStickyItems
            _oldMassesOfStickyItems.Add(childName,childMass);
            // extract the Collider component from the collision object
with GetContact(...)
            Collider col = collision.GetContact(0).thisCollider;
            // set the parent of the collided object
            child.transform.parent = col.GetComponent<Transform>();
            // update the Katamari mass
            UpdateKatamariMass(childMass,"add");
            // after merging the ball with the collided object, it is
necessary to destroy the RigidBody component to facilitate their movement
as a singular entity
            Destroy(child.GetComponent<Rigidbody>());
        }
    }

    /// <summary>
    /// A helper function to get katamari ball's current mass.
    /// </summary>
    /// <returns>Mass of the katamari ball</returns>
    private float GetKatamariMass()
    {
        return gameObject.GetComponent<Rigidbody>().mass;
    }

    /// <summary>
    /// A helper function to change katamari ball's mass.
    /// </summary>
    /// <param name="mass">Mass to be added to katamari ball.</param>
```

```
    /// <param name="operation">Determines the type of operation that will
be performed (substraction or addition)</param>
    private void UpdateKatamariMass(float mass, string operation)
    {
        if (operation == "add")
            gameObject.GetComponent<Rigidbody>().mass += mass;
        else if (operation == "substract")
            gameObject.GetComponent<Rigidbody>().mass -= mass;
        else
            Debug.Log("Error while doing operation.");
    }
}
```

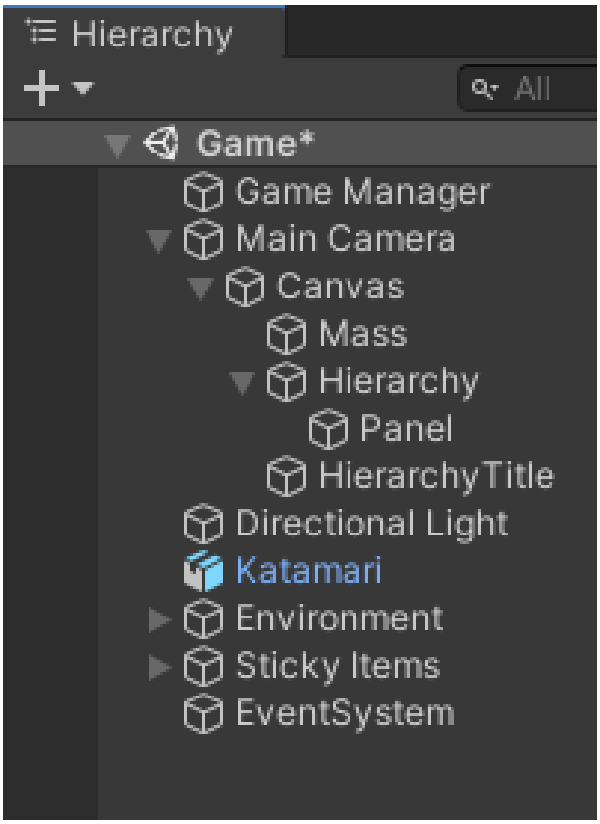
Opis skripte PlayerInformation.cs

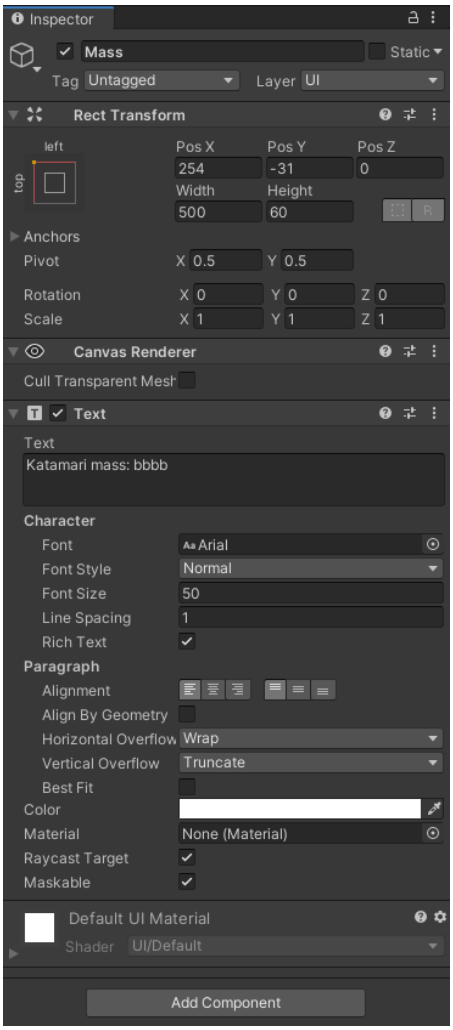
Unutar funkcije Update() definiran je objekt mass tipa float unutar kojeg se dohvaća i sprema masa objekta koji ima ocu skriptu kao komponentu, a to je Katamari. String help sadrži tekst koji se treba prikazivati na zaslonu „Katamari mass: “. Dohvaćamo komponentu text objekt _katamariMass i mijenjamo vrijednost u help + mass.ToString(). Zatim selekcijom if() provjeravamo je li plaerPrefs vrijednost oznake „print“ jednaka 1, te ako je ulazimo u tijelo selekcije. Unutar tijela selekcije stvara se lista stringova visited. Komponenta text objekta _hierarchy se postavlja na prazno, kako bi se maknula prijašnja hijerarhija koja je bila prikazana na zaslonu. U niz allChildren tipa Transform dohvaćaju se sve komponente Transform sve djece objekta _katamari. Komponenta text objekta _hierarchy postavlja se u vrijednost komponente name od prvog objekta iz niza. Zatim se odvija poziv funkcije PrintHelp s predanim parametrima allChildren, visited i brojem 1. Nakon poziva funkcije PlayerPrefs postavlja vrijednost oznake „print“ na 0. Funkcija PrintHelp odvija prolazak kroz hijerarhiju rekurzivno. Iako se unutar niza nalaze sva djeca, ne može se jednostavno odrediti daljnje grananje. Rekurzivni poziv omogućava kretanje do listova, te praćenje posjećenih, odnosno ispisani objekata iz hijerarhije. Unutar funkcije nalazi se foreach petlja koja prolazi po svim elementima niza allChildren. Nadalje selekcija if() provjerava nalazi li se trenutni element u listi visited te da odabrani element nije prvi element iz liste jer smo njega već ispisali u prijašnjem prolasku. Ako se ne nalazi u listi i nije prvi element niza, izvršava se tijelo selekcije. Unutar element se dodaje u listu visited. Zatim se dodaje prelazak u novi red te se stvara pomak. Unutar petlje for koja se kreće od 0 do vrijednosti parametra depth, stringu pomak dodaje se string dvostruka praznina. Na taj način svaka slijedeća dubina u hijerarhiji će imati 2 praznine više i na taj način vizualno tvoriti hijerarhiju. Nakon petlje komponenta text objekta _hierarchy se dodaje prijašnji sadržaj komponente, pomak, znak je veće i ime objekta. Zatim se pomoću selekcije provjerava ima li trenutni objekt podređene objekte. Ako ima izvršava se tijelo selekcije, odnosno stvara se novi niz newChildren s novim komponentama. Niz newChildren će sadržavati sve podređene elemente tog objekta i sami taj objekt. Zatim se rekuzivno poziva funkcija s parametrima newChildren, visited i depth+1. Na taj način smo osigurali da se dalje prenosi ista lista kako ne bi izgubili posjećene elemente i da se dubina povećava s daljnjim pronalaskom podređenih elemenata.

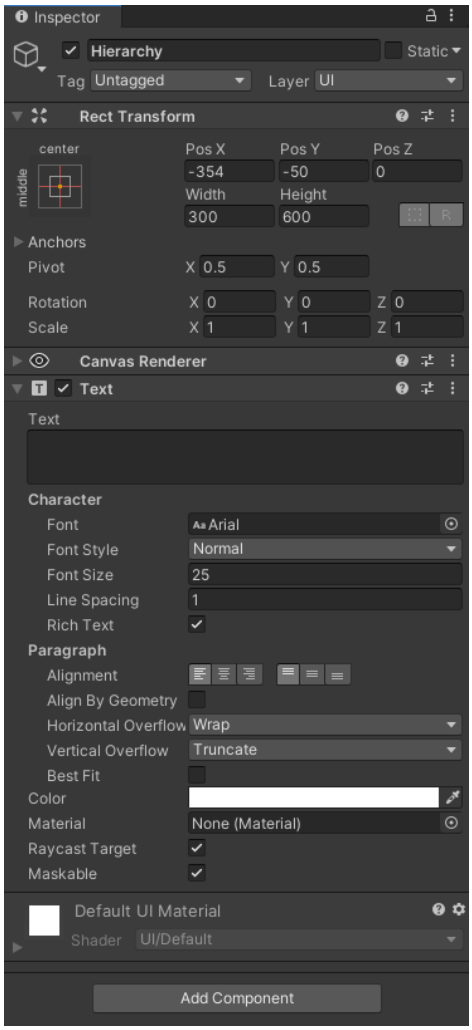
Skripta PlayerInformation.cs

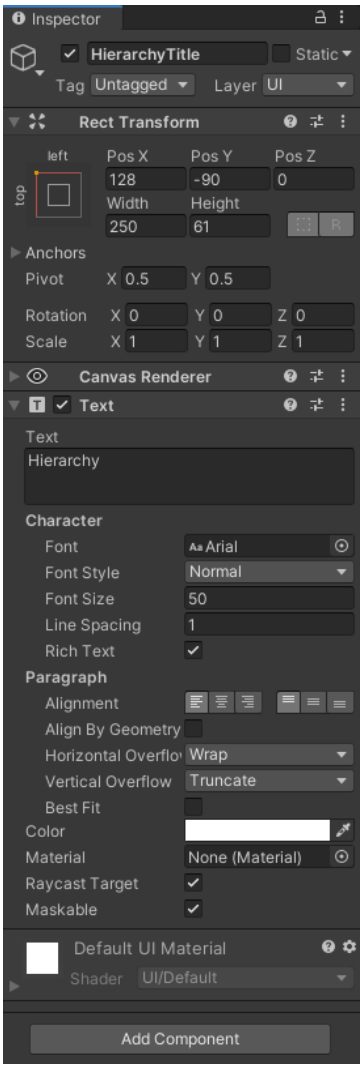
```
using UnityEngine;
using UnityEngine.UI;
using System.Collections.Generic;

namespace Assets.Scripts
{
    /// <summary>
    /// Class for viewing information about hierarchy and katamari mass
    /// </summary>
    public class PlayerInformation : MonoBehaviour
    {
        [SerializeField] private Text _katamariMass = null;
        [SerializeField] private Text _hierarchy = null;
        private Rigidbody _katamari = null;
        private Renderer [] _renderers = null;

        /// <summary>
        /// Start is called before the first frame update.
        /// Initialize Rigidbody component.
        /// </summary>
        private void Start()
        {
            _katamari = GetComponent<Rigidbody>();
        }

        /// <summary>
        /// Update is called once per frame.
        /// Get rendered objects and update the view of hierarchy and
katamari ball mass
        /// </summary>
        private void Update()
        {
            float mass = gameObject.GetComponent<Rigidbody>().mass;
            string help = "Katamari mass: ";
            _katamariMass.text = help + mass.ToString();

            if (PlayerPrefs.GetInt("print")==1)
            {
                List<string> visited = new List<string>();
                _hierarchy.text = "";

                Transform[] allChildren =
_katamari.GetComponentsInChildren<Transform>();
                _hierarchy.text = _hierarchy.text + ">" +
allChildren[0].name;
                PrintHelp(allChildren, visited, 1);
                PlayerPrefs.SetInt("print", 0);
            }
        }

        private void PrintHelp(Transform[] allChildren, List<string>
visited, int depth)
        {
            foreach (Transform child in allChildren)
```

```
            {
                if
(!visited.Contains(child.gameObject.name)&&child!=allChildren[0])
                {
                    visited.Add(child.gameObject.name);
                    _hierarchy.text = _hierarchy.text + "\n";
                    string pomak="";
                    for(int i=0;i<depth; i++)
                    {
                        pomak = pomak + "   ";
                    }

                    _hierarchy.text = _hierarchy.text + pomak + ">" +
child.gameObject.name;
                    if (child.childCount > 0)
                    {
                        Transform[] newChildren =
child.gameObject.GetComponentsInChildren<Transform>();
                        PrintHelp(newChildren, visited, depth+1);
                    }
                }
            }
        }

    }
}
```



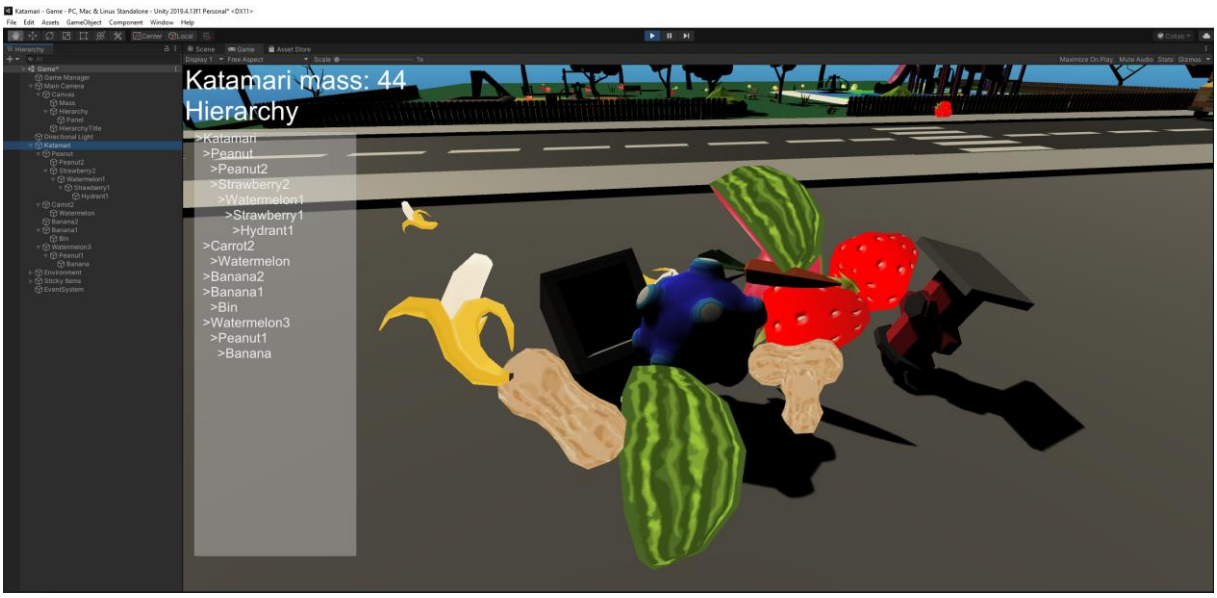

Inspector
Mass
Static
Tag Untagged
Layer UI
Rect Transform
left
top
Pos X 254
Pos Y -31
Pos Z 0
Width 500
Height 60
Anchors
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
Cull Transparent Mesh
Text
Text
Katamari mass: bbbb
Character
Font Arial
Font Style Normal
Font Size 50
Line Spacing 1
Rich Text
Paragraph
Alignment
Align By Geometry
Horizontal Overflow Wrap
Vertical Overflow Truncate
Best Fit
Color
Material None (Material)
Raycast Target
Maskable
Default UI Material
Shader UI/Default
Add Component

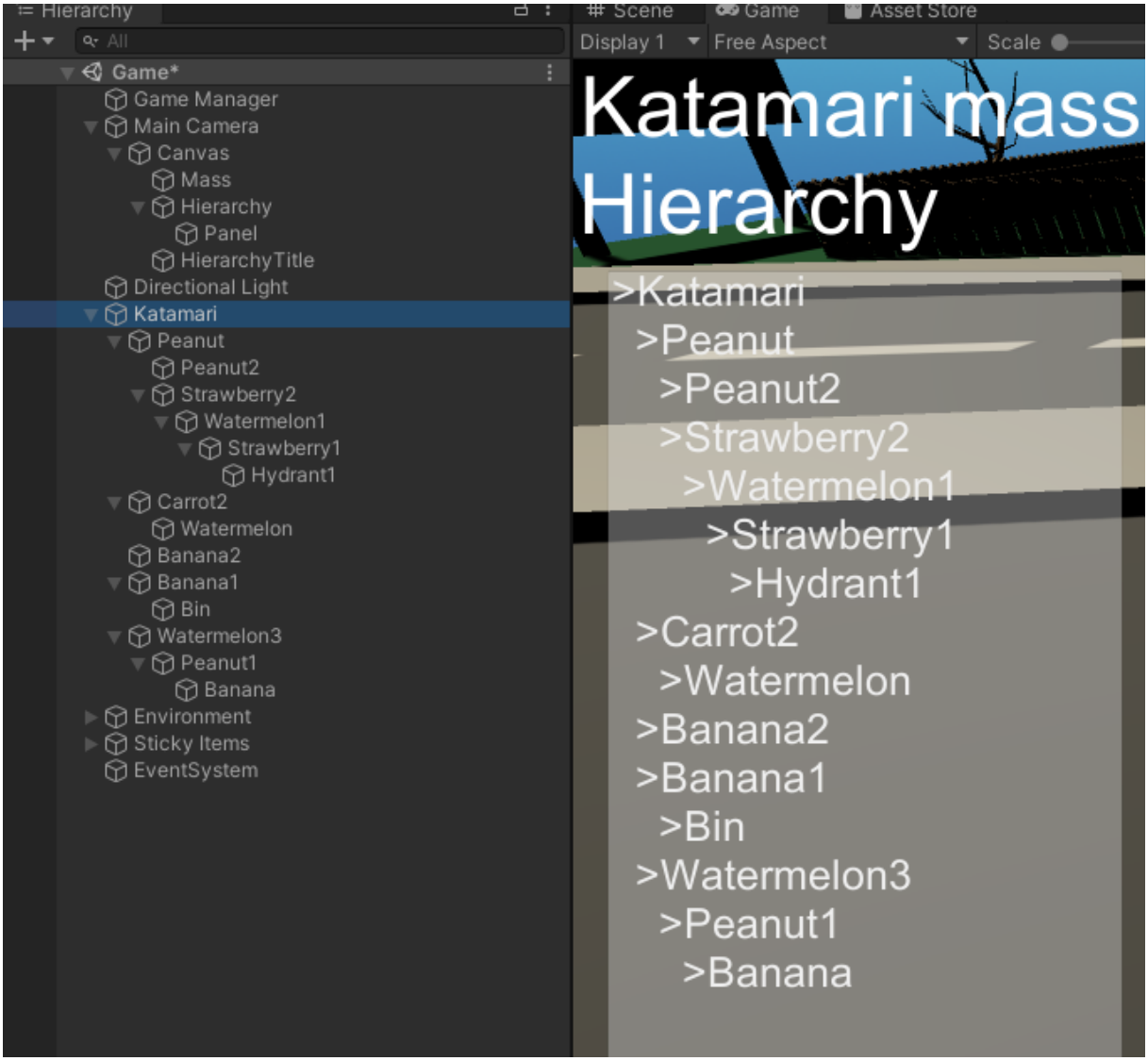

Inspector
Hierarchy
Static
Tag Untagged
Layer UI
Rect Transform
center
middle
Pos X -354
Pos Y -50
Pos Z 0
Width 300
Height 600
Anchors
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
Text
Text
Character
Font Arial
Font Style Normal
Font Size 25
Line Spacing 1
Rich Text
Paragraph
Alignment
Align By Geometry
Horizontal Overflow Wrap
Vertical Overflow Truncate
Best Fit
Color
Material None (Material)
Raycast Target
Maskable
Default UI Material
Shader UI/Default
Add Component


Inspector
HierarchyTitle
Static
Tag Untagged
Layer UI
Rect Transform
left
top
Pos X 128
Pos Y -90
Pos Z 0
Width 250
Height 61
Anchors
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
Text
Text
Hierarchy
Character
Font Arial
Font Style Normal
Font Size 50
Line Spacing 1
Rich Text
Paragraph
Alignment
Align By Geometry
Horizontal Overflo Wrap
Vertical Overflow Truncate
Best Fit
Color
Material None (Material)
Raycast Target
Maskable
Default UI Material
Shader UI/Default
Add Component


Inspector
Katamari
Static
Tag Player
Layer Default
Model Open Select Overrides
Transform
Icosphere (Mesh Filter)
Mesh Renderer
Rigidbody
Mass 1
Drag 0
Angular Drag 0.05
Use Gravity
Is Kinematic
Interpolate None
Collision Detection Discrete
Constraints
Freeze Position X Y Z
Freeze Rotation X Y Z
Info
Mesh Collider
Convex
Is Trigger
Cooking Options Mixed...
Material None (Physic Material)
Mesh Icosphere
Player Movement (Script)
Player Collision (Script)
Script PlayerCollision
Sticky Items Sticky Items
Player Information (Script)
Script PlayerInformation
Katamari Mass Mass (Text)
Hierarchy Hierarchy (Text)


Katamari mass: 1
Hierarchy
>Katamari


Katamari mass: 44
Hierarchy
>Katamari
>Peanut
>Peanut2
>Strawberry2
>Watermelon1
>Strawberry1
>Hydrant1
>Carrot2
>Watermelon
>Banana2
>Banana1
>Bin
>Watermelon3
>Peanut1
>Banana


Hierarchy
All
Game*
Game Manager
Main Camera
Canvas
Mass
Hierarchy
Panel
HierarchyTitle
Directional Light
Katamari
Peanut
Peanut2
Strawberry2
Watermelon1
Strawberry1
Hydrant1
Carrot2
Watermelon
Banana2
Banana1
Bin
Watermelon3
Peanut1
Banana
Environment
Sticky Items
EventSystem
Scene
Game
Asset Store
Display 1
Free Aspect
Scale
Katamari mass
Hierarchy
>Katamari
>Peanut
>Peanut2
>Strawberry2
>Watermelon1
>Strawberry1
>Hydrant1
>Carrot2
>Watermelon
>Banana2
>Banana1
>Bin
>Watermelon3
>Peanut1
>Banana