**SolarSystem**

**Cuprins**

1. Cuprins 1
2. Informații Generale 2
3. Settings 3
4. TimeMultiplier Class 3
5. Scale Class 6
6. ShowMoons Class 8
7. ShowOrbit Class 10
8. ShowInfoText Class 12
9. PlayButton Class 14
10. InputManager 16
11. BackButton Class 16
12. FullScreenMode Class 18
13. SwitchCamera Class 20
14. SelectMenuInput Class 26
15. Generator Class 28
16. ObserverSelector Class 32
17. TargetSelector Class 34
18. SolarSystem 36
19. Initialize Class 36
20. Gravity Class 41
21. CelestialObject Class 44
22. DayCounter Class 48
23. HotKeys 49
24. Videoclipuri/Poze 50

**Informații generale**

**De ce Unity?**

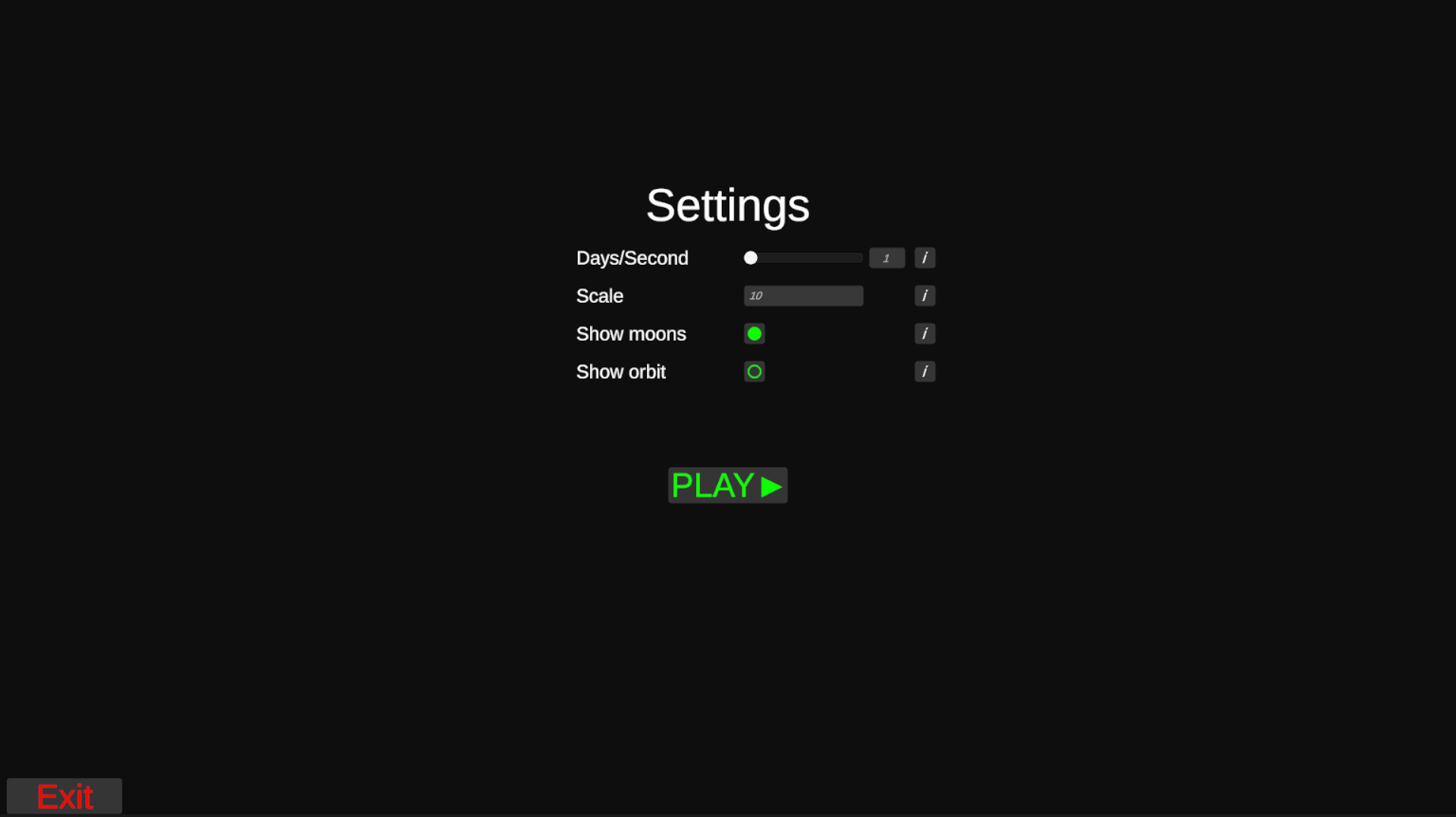
* Cunoașterea interfaței, sintaxei și modului de lucru.
* Aprofundarea cunoștințelor de game development.
* Implementarea ușoara a fenomenelor fizice cu precizie mare.

**De ce C#?**

* Limbajul de programare principal Unity.
* Limbaj de programare orientat pe obiecte.
* Viteza mare de compilare.

**Bibliotecile utilizate:**

* [System.Collections](https://docs.unity3d.com/Packages/com.unity.collections@0.0/manual/index.html)
* [System.Collections.Generic](https://learn.microsoft.com/en-us/dotnet/api/system.collections.generic?view=net-8.0)
* UnityEngine
* [UnityEngine.UI](https://docs.unity3d.com/Packages/com.unity.ugui@1.0/api/UnityEngine.UI.html)
* [TMPro](https://docs.unity3d.com/Packages/com.unity.textmeshpro@1.3/api/TMPro.html)
* [UnityEngine.SceneManagement](https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.html)
* [MonoBehaviour](https://docs.unity3d.com/ScriptReference/MonoBehaviour.html)

**Settings**

ShowInfoText

PlayButton

BackButton

TimeMultiplier

Scale

ShowMoons

ShowOrbit

TimeMultiplier Class

**Utilizare:**

* Permite alegerea nivelului vitezei trecerii timpului.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using UnityEngine.UI;

public class TimeMultiplier : MonoBehaviour

1. **Variabilele clasei:**
2. public TMP\_InputField inputField;
3. public Slider slider;
4. public ShowMoons showMoons;
5. int[] intArray;
6. public int timeMultiplierValue = 1;
7. Referință către un *câmp de intrare* de tip text.
8. Referință către un obiect de tip *cursor*.
9. Referință către un obiect de clasă *ShowMoons*.
10. Un vector cu elemente de tip int.
11. Valoarea variabilei *timeMultiplier*.
12. **Funcția OnValueChangedInputField:**

public void OnValueChangedInputField()

{

1. intArray = inputField.text.ToIntArray();
2. for (int i = 0; i < intArray.Length; i++)

if (intArray[i]< '0'||intArray[i]> '9'|| intArray.Length > 2)

{

inputField.text = "";

break;

}

}

* Funcția este apelată la fiecare input de la tastatură efectuată în zona de *câmp de intrare*.

1. Textul introdus de la tastatură este transformat într-un vector de tip int cu codurile ASCII ale textului.
2. Se verifica dacă s-au introdus doar numere și dacă lungimea textului este de maxim 2 caractere.
3. **Funcția OnValueChangedSlider:**

public void OnValueChangedSlider()

{

inputField.text = slider.value.ToString();

OnEndEdit();

}

* Funcția este apelată la fiecare schimbare de valoare a *cursorului*.
* Textul *câmpului de intrare* este modificat și se apelează funcția *OnEndEdit*.

1. **Funcția OnEndEdit:**

public void OnEndEdit()

{

1. timeMultiplierValue = 0;

intArray = inputField.text.ToIntArray();

1. for (int i = 0; i < intArray.Length; i++)

timeMultiplierValue = timeMultiplierValue\*10+intArray[i]-'0';

1. if(showMoons.value&&timeMultiplierValue>12)

{

timeMultiplierValue = 12;

inputField.text = "12";

}

else

showMoons.timeMultiplierValue = timeMultiplierValue;

1. if (timeMultiplierValue > 50)

{

timeMultiplierValue = 50;

inputField.text = "50";

}

if(timeMultiplierValue<1)

{

timeMultiplierValue = 1;

inputField.text = "1";

}

1. slider.value = timeMultiplierValue;

}

1. Se inițializează valoarea variabilei *timeMultiplier* și a vectorului *intArray*.
2. Se calculează valoarea variabilei *timeMultiplier*.
3. Dacă simularea permite afișarea sateliților naturali a planetelor și valoarea variabilei *timeMultiplier* este mai mare decat 12, aceasta este setată la 12. Altfel, se salvează valoarea.
4. Se verifică cazul .
5. Se updatează valoarea cursorului.

Scale Class

**Utilizare:**

* Permite alegerea lungimii scalei care influențeaza direct cât de multe corpuri cerești sunt vizibile în simulare.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using UnityEngine.UI;

public class Scale : MonoBehaviour

1. **Variabilele clasei:**
2. public TMP\_InputField inputField;
3. public int scale=10;
4. int[] intArray;
5. Referință către un *câmp de intrare* de tip text.
6. Variabila *scalei*(**default scale=10**).
7. Un vector cu elemente de tip int.
8. **Funcția OnValueChanged:**

public void OnValueChanged()

{

intArray = inputField.text.ToIntArray();

for (int i = 0; i < intArray.Length; i++)

if (intArray[i] <'0' || intArray[i]>'9' || intArray.Length>2)

{

inputField.text = "";

break;

}

}

* Funcția este identică cu *OnValueChangedInputField* din clasa *TimeMultiplier*.

1. **Funcția OnEndEdit:**

public void OnEndEdit()

{

1. scale = 0;

for (int i = 0; i < intArray.Length; i++)

scale = scale \* 10 + intArray[i] - '0';

1. if (scale > 30)

{

scale = 30;

inputField.text = "30";

}

if (scale < 1)

{

scale = 1;

inputField.text = "1";

}

}

1. Se calculează valoarea scalei.
2. Se verifică cazul .

ShowMoons Class

**Utilizare:**

* Permite alegerea afișarii sateliților naturali.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using UnityEngine.UI;

public class ShowMoons : MonoBehaviour

1. **Variabilele clasei:**
2. public TMP\_Text buttonText;
3. public TimeMultiplier timeMultiplier;
4. public int timeMultiplierValue;
5. public bool value=true;
6. Referință către zona de text a unui buton.
7. Referință către un obiect de clasă *TimeMultiplier*.
8. Valoarea variabilei *timeMultiplier*.
9. Variabila care definește dacă se afișeaza sateliții naturali(**default=true**).
10. **Funcția OnButtonClick:**

public void OnButtonClick()

{

1. value = !value;
2. if(value)

{

buttonText.color = Color.green;

buttonText.text = "●";

1. if(timeMultiplier.timeMultiplierValue>12)

{

timeMultiplierValue = timeMultiplier.timeMultiplierValue;

timeMultiplier.inputField.text = "12";

timeMultiplier.OnEndEdit();

}

}

1. else

{

buttonText.color = Color.red;

buttonText.text = "◌";

timeMultiplier.inputField.text = timeMultiplierValue.ToString();

timeMultiplier.OnEndEdit();

}

}

1. Se schimbă valoarea variabilei.
2. În cazul în care aceasta este adevarată textul butonului se modifică, iar culoarea sa va fi verde.
3. Dacă valoarea variabilei *timeMultiplier* este mai mare decât *12* aceasta se salvează, apoi modifică textul și valoarea variabilei *timeMultiplier*.
4. În cazul în care variabila este falsă textul butonului se modifica, iar culoarea sa va fi rosu. Textul și valoarea variabilei *timeMultiplier* se resetează cu ultima s-a valoare înainte de modificari.

ShowOrbit Class

**Utilizare:**

* Permite afisarea oribitei corpurilor cerestre.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using UnityEngine.UI;

public class ShowOrbit : MonoBehaviour

1. **Variabilele clasei:**
2. public TMP\_Text buttonText;
3. public bool value = true;
4. Referință către un obiect de tip text.
5. Valoarea de adevăr a afișarii orbitei.
6. **Funcția OnButtonClick:**

public void OnButtonClick()

{

1. value = !value;
2. if(value)

{

buttonText.text = "○";

buttonText.color = Color.green;

}

else

{

buttonText.text = "◌";

buttonText.color = Color.red;

}

}

1. Se schimbă valoarea de adevar.
2. În cazul în care valoarea este adevarată se modifică textul și culoarea in verde, altfel în rosu.

ShowInfoText Class

**Utilizare:**

* Este un text informativ care are scopul de a explica cum va influența fiecare parametru simularea.
* Clasa este atașată unor obiecte de tip *Button*.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using UnityEngine.UI;

public class ShowInfoText : MonoBehaviour

1. **Variabilele clasei:**
2. public List<ShowInfoText> showInfoTextsList = new List<ShowInfoText>();
3. public TMP\_Text infoText;
4. bool showInfoText = false;
5. public string text;
6. Lista cu elemete de tip *ShowInfoText*.
7. Referință la o un obiect de tip text.
8. Variabila care definește afișarea textului.
9. Șirul de caractere care conține textul predefinit.
10. **Funcția OnButtonClick:**

public void OnButtonClick()

{

1. showInfoTextsList.AddRange(FindObjectsOfType<ShowInfoText>());
2. for (int i = 0; i < showInfoTextsList.Count; i++)

if(this!=showInfoTextsList[i])

showInfoTextsList[i].showInfoText = false;

1. showInfoText = !showInfoText;
2. infoText.text = text;

infoText.gameObject.SetActive(showInfoText);

infoText.gameObject.transform.position = this.transform.position + new Vector3(120,-10,0);

1. showInfoTextsList.Clear();

showInfoTextsList.TrimExcess();

}

1. Se adaugă toate elementele de tip *ShowInfoText* în listă.
2. Toate elementele listei diferite de *butonul* activat caruia îi este atașată clasa *ShowInfoText* devin inactive.
3. Se schimbă valoarea de adevăr a afișării textului.
4. Se setează parametrii textului: textul de afișat, starea și poziția câmpului de text.
5. Lista este golită.

PlayButton Class

**Utilizare:**

* Pornește simularea dupa ce au fost efectuate setarile dorite.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class PlayButton : MonoBehaviour

1. **Variabilele clasei:**
2. public Scale scale;
3. public ShowMoons showMoons;
4. public ShowOrbit showOrbit;
5. public TimeMultiplier timeMultiplier;
6. Referință către un obiect de clasă *Scale*.
7. Referință către un obiect de clasă *ShowMoons*.
8. Referință către un obiect de clasă *ShowOrbit*.
9. Referință către un obiect de clasă *TimeMultiplier*.
10. **Funcția OnButtonClick:**

public void OnButtonClick()

{

1. Initialize.SetScale(scale.scale);

Initialize.SetShowMoonsValue(showMoons.value);

Initialize.SetShowOrbitValue(showOrbit.value);

Initialize.SetTimeMultiplier(timeMultiplier.timeMultiplierValue\* 3600\*24);

1. SceneManager.LoadScene("SolarSystem");

}

1. Se inițializează variabilele din clasa *Initialize* cu valorile specifice din meniul *Settings*.
2. Se deschide șcena *SolarSystem*.

**InputManager**

BackButton Class

**Utilizare:**

* Permite revenirea la meniul *Settings* sau închiderea aplicației.
* Funcționalitatea butonului poate fi accesată atât prin butonul ***Back*** cât și prin apasarea tastei ***Escape*.**

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.SceneManagement;

public class BackButton : MonoBehaviour

1. **Funcția Update:**

public void Update()

{

if (Input.GetKeyDown(KeyCode.Escape))

OnClick();

}

* Se verifică dacă tasta *Escape* este apăsată.

1. **Funcția OnClick**

public void OnClick()

{

1. if(SceneManager.GetActiveScene()==

SceneManager.GetSceneByName("Settings"))

Application.Quit(0);

1. if(SceneManager.GetActiveScene()==

SceneManager.GetSceneByName("SolarSystem"))

SceneManager.LoadScene("Settings");

}

1. Dacă este deschis meniul Settings, aplicația se va închide.
2. Dacă este deschisă simularea, butonul va permite întoarcerea la meniul Settings.

FullScreenMode Class

**Utilizare:**

* Permite trecerea din modul Fullscreen in modul Window.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class FullScreenMode : MonoBehaviour

1. **Variabilele clasei:**

public static bool fullScreenMode=true;

* Indică valoarea de adevar a modului Fulscreen(default:**true**).

1. **Funcția Start**

public void Start()

{

Screen.fullScreen = fullScreenMode;

}

* Se activează modul Fullscreen în funcție de valoarea de adevăr a variabilei *fullScreenMode*.

1. **Funcția Update:**

public void Update ()

{

1. if (Input.GetKeyDown(KeyCode.F11))

{

1. fullScreenMode = !fullScreenMode;

Screen.fullScreen = fullScreenMode;

}

}

1. Se verifică dacă tasta *F11* este apăsată.
2. Se neagă valoarea de adevăr a variabilei *fullScreenMode* și modul Fullscren primește această valoare, astfel se trece de la Fullscreen la Window și invers.

SwitchCamera Class

**Utilizare:**

* Permite vizualizarea mai ușoară a sistemului, orientarea catre un anumit corp cerestru și observarea mai ușoara a sateliților naturali.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using UnityEngine.UI;

public class SwitchCamera : MonoBehaviour

1. **Variabilele clasei:**
2. public Camera mainCamera;
3. public int cameraObserverIndex = 0;
4. public int cameraTargetIndex = 0;
5. public int maxCameraIndex;
6. public List<CelestialObject> celestialObjects;
7. public GameObject observer;
8. public GameObject target;
9. public TMP\_Text obs;
10. public TMP\_Text targ;
11. public bool update = false;
12. public int updateCount = 0;
13. int oldTimeMultiplier;
14. int oldScale;
15. Referință către camera principala.
16. Indexul corpului cerestru de pe care se fac observațiile.
17. Indexul corpului cerestru țintă.
18. Indexul maxim al camerei.
19. Lista obiectelor de tipul *CelestialObject*.
20. Referință către corpul de pe care se fac observațiile.
21. Referință către corpul țintă.
22. Referință către zona text a observatorului.
23. Referință către zona text a țintei.
24. Valoarea de adevăr a update-ului.
25. Numărul de frame-uri care au trecut de la ultimul update.
26. Valoarea precedentă a variabilei *timeMultiplier*.
27. Valoarea precedentă a variabilei *scale*.
28. **Funcția Start:**

private void Start()

{

1. celestialObjects = Initialize.GetCelestialObjects();
2. maxCameraIndex = Initialize.GetMaxCameraIndex();
3. oldTimeMultiplier = Initialize.GetTimeMultiplier();
4. oldScale = Initialize.GetScale();

}

1. Se inițializează lista corpurilor cerestre.
2. Se inițializează indexul maxim de cameră.
3. Se stochează valoarea inițială a variabilei *timeMultiplier*.
4. Se stochează valoarea inițiala a variabilei *scale*.
5. **Funcția Update:**

public void Update()

{

1. if (update && updateCount < 10)

{

updateCount++;

}

1. else if (updateCount == 10)

{

for (int i = 0; i < celestialObjects.Count; i++)

if (celestialObjects[i].showTrail)

celestialObjects[i].trail.time = celestialObjects[i].trailTime;

update = false;

updateCount = 0;

}

1. int input = 0;

input = Input.GetKeyDown(KeyCode.RightArrow) ? 1 : input;

input = Input.GetKeyDown(KeyCode.LeftArrow) ? 2 : input;

input = Input.GetKeyDown(KeyCode.UpArrow) ? 3 : input;

input = Input.GetKeyDown(KeyCode.DownArrow) ? 4 : input;

1. switch(input)

{

1. case 1:
2. celestialObjects[cameraObserverIndex].trail.time = celestialObjects[cameraObserverIndex].trailTime;

celestialObjects[cameraObserverIndex].showTrail = true;

1. cameraObserverIndex++;

if (cameraObserverIndex > maxCameraIndex)

cameraObserverIndex = 0;

break;

case 2:

celestialObjects[cameraObserverIndex].trail.time = celestialObjects[cameraObserverIndex].trailTime;

cameraObserverIndex--;

if (cameraObserverIndex < 0)

cameraObserverIndex = maxCameraIndex;

break;

1. case 3:
2. cameraTargetIndex++;

if (cameraTargetIndex > maxCameraIndex)

cameraTargetIndex = 0;

break;

case 4:

cameraTargetIndex--;

if (cameraTargetIndex < 0)

cameraTargetIndex = maxCameraIndex;

break;

}

1. if(input!=0)

{

1. observer = celestialObjects[cameraObserverIndex].gameObject;
2. celestialObjects[cameraObserverIndex].trail.time = 0;

celestialObjects[cameraObserverIndex].showTrail = false;

1. obs.text = "Obs:" + observer.name;
2. target = celestialObjects[cameraTargetIndex].gameObject;
3. targ.text = "Targ:" + target.name;
4. UpdateScale(observer.transform.position.magnitude, target.transform.position.magnitude);

}

1. mainCamera.transform.position = observer.transform.position + new Vector3(0, 0, (cameraObserverIndex == cameraTargetIndex) ? (celestialObjects[cameraObserverIndex].objectType == CelestialObject.CelestialObjectType.Sun ? 100 : 50) - observer.transform.position.z : 0);
2. transform.LookAt(target.transform.position, Vector3.forward);

}

1. Dacă *update* este activ(true) se incrementează *updateCount* în fiecare frame până la valoarea *10*.
2. Când *updateCount* ajunge la valoarea *10* se resetează timpul orbitei pentru corpurile cerestre care permit afișarea orbitei. Update-ul se încheie, iar numararea se resetează.
3. Este setată valoarea variabilei *input* în funcție de comenzile de la tastatură:
4. În funcție de valoarea *input* se execută urmatoarele comenzi:
5. Case 1/Case 2(incrementare/decrementare *observator*).
6. Vechiul observator primeste valorile inițiale.
7. Se incrementează/decrementează indexul observatorului. In cazul in care se atinge indexul maxim+1/-1 valoarea indexului observatorului devine 0/maxim.
8. Case 3/Case 4(incrementare/decrementare *țintă*).
9. Se incrementează/decrementează indexul țintei. In cazul în care se atinge indexul maxim+1/-1 valoarea indexului țintei devine 0/maxim.
10. Dacă s-a primit un input valabil de la tastatură se efectuează urmatoarele comenzi:
11. Se updatează observatorul.
12. Se oprește afișarea orbitei sale.
13. Se updatează textul în funcție de numele observatorului.
14. Se updatează ținta.
15. Se updatează textul în funcție de numele țintei.
16. Este apelată funcția *UpdateScale* cu valoarea poziției observatorului și țintei.
17. Ân fiecare frame se updatează poziția camerei astfel:

* -poziția camerei
* -poziția observatorului
* -vector auxiliar
* -corpul cerestru **observator**
* -corpul cerestru **țintă**
* -componenta *z* a vectorului

1. Rotește camera astfel încât să fie orientată spre corpul ceresc țintă.
2. **Funcția UdateScale:**

public void UpdateScale(float r1, float r2)

{

if (Mathf.Abs(r1 - r2) < 0.5 && Mathf.Abs(r1 - r2) > 0.0005 && Initialize.GetScale() != 1000)

{

UpdateCelestialObjects(1000, 3600 \* 12);

update = true;

}

if (Mathf.Abs(r1 - r2) > 500 && Initialize.GetScale() != oldScale)

{

UpdateCelestialObjects(oldScale, oldTimeMultiplier);

update = true;

}

if (Mathf.Abs(r1 - r2) < 0.0005 && Mathf.Abs(r1 - r2) !=0 && Initialize.GetScale() != 10000)

{

UpdateCelestialObjects(10000, 3600);

update = true;

}

}

* În funcție de se modifică valoarea variabilelor *scale* și *timeMultiplier* cu ajutorul funcției *UpdateCelestialObjects*. În cazul în care trebuie efectuate modificari *update*=true.

1. **Funcția UpdateCelestialObjects:**

void UpdateCelestialObjects(int newScale, int newTimeMultiplier)

{

1. Initialize.SetTimeMultiplier(newTimeMultiplier);

for (int i = 0; i < celestialObjects.Count; i++)

{

1. celestialObjects[i].trail.time = 0;
2. celestialObjects[i].transform.position \*= (float)newScale / Initialize.GetScale();
3. celestialObjects[i].trailTime = celestialObjects[i].T \* 3600f \* 24f / newTimeMultiplier;

}

1. Initialize.SetScale(newScale);

}

1. Se inițializează variabila *timeMultiplier* cu noua sa valoare.
2. Orbita corpurilor cerestre devine inactivă.
3. Poziția corpurilor cerești este modificată în funție de noua valoare a scalei:

* -noua poziție a corpului
* -vechea poziție a corpului
* -valoarea nouă a scalei
* -valoarea veche a scalei

1. Perioada orbitală este modificată în funcție de noua valoare a variabilei *timeMultiplier*.
2. Se inițializează valoarea scalei cu noua valoare.

SelectMenuInput Class

**Utilizare:**

* Deschide meniul de selectare a *observatorului* și *țintei*.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class SelectMenuInput : MonoBehaviour

1. **Variabilele clasei:**
2. public GameObject obsScr, targScr, obsTxt, targTxt;
3. public bool setActive = false;
4. Referință la 4 obiecte setate in fereastra *Inspector* din Unity.
5. Starea meniului *activ/inactiv*(**default=false**).
6. **Funcția Update:**

public void Update()

{

if (Input.GetKeyDown(KeyCode.Space))

{

setActive = !setActive;

obsScr.SetActive(setActive);

obsTxt.SetActive(setActive);

targScr.SetActive(setActive);

targTxt.SetActive(setActive);

}

}

* În momentul apăsării tastei *Space* valoarea de adevar a variabilei *setActive* este schimbată, iar cele 4 obiecte sunt activate/dezactivate în funcție de această variabila.

trgTxt

obsTxt

ButtonSoare Obs

****

ButtonSoare Trg

trgSrc

obsSrc

Generator Class

**Utilizare:**

* Permite navigarea mai rapida de la un corp cerestru la altul si alegerea observatorului si tintei.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using UnityEngine.UI;

public class Generator : MonoBehaviour

1. **Variabilele clasei:**
2. public enum TypeOfGenerator {Observer,Target};
3. public TypeOfGenerator generatorType;
4. public GameObject content;
5. List<CelestialObject> celestialObjects = new List<CelestialObject>();
6. public SwitchCamera switchCamera;
7. public List<GameObject> button = new List<GameObject>();
8. public Sprite sprite;
9. Lista de tipuri de generare.
10. Variabila specifică tipului de generare.
11. Referință la zona de generare.
12. Lista corpurilor cerești.
13. Referință la un obiect de tip *SwitchCamera*.
14. Lista de obiecte de tip buton.
15. Referință la un obiect grafic 2D.
16. **Funcția Start:**

public void Start()

{

1. celestialObjects = Initialize.GetCelestialObjects();

for(int i=0;i<celestialObjects.Count;i++)

{

1. GameObject obj = new GameObject("Button" + celestialObjects[i].name);

button.Add(obj);

1. button[i].transform.SetParent(content.transform);
2. GameObject newText = new GameObject("Text" + celestialObjects[i].name);

newText.AddComponent<TextMeshProUGUI>();

1. TextSetup(newText.GetComponent<TextMeshProUGUI>(), celestialObjects[i].name);

newText.transform.SetParent(obj.transform);

1. obj.AddComponent<Image>();

ImageSetup(obj.GetComponent<Image>(),i);

1. obj.AddComponent<Button>();
2. if(generatorType==TypeOfGenerator.Observer)

{

obj.AddComponent<ObserverSelector>();

obj.GetComponent<ObserverSelector>().switchCamera = FindObjectOfType<SwitchCamera>();

obj.GetComponent<ObserverSelector>().ordIndex = i;

obj.GetComponent<Button>().onClick.AddListener(obj.GetComponent<ObserverSelector>().OnClickEvent);

}

else

{

obj.AddComponent<TargetSeletor>();

obj.GetComponent<TargetSeletor>().switchCamera = FindObjectOfType<SwitchCamera>();

obj.GetComponent<TargetSeletor>().ordIndex = i;

obj.GetComponent<Button>().onClick.AddListener(obj.GetComponent<TargetSeletor>().OnClickEvent);

}

}

1. content.GetComponent<RectTransform>().sizeDelta= new Vector2(0, celestialObjects.Count \* 30+5);

}

1. Se inițializează lista corpurilor cerești.
2. Pentru fiecare corp ceresc se creează un nou obiect. Și se adaugă listei de butoane.
3. Se setează *parintele* butonului.
4. Se crează un nou obiect caruia i se atribuie componenta *TextMesh*.
5. Se apelează funcția de setare a textului care are ca parametrii componenta textului și numele corpului ceresc. Apoi este setat parintele.
6. Se adaugă componenta *imagine* și se apelează funcția de setare a imaginii care are ca parametrii componenta imagine si indexul corpului cerestru.
7. Se adaugă componenta *buton*.
8. Dacă tipul de generare este *Observer* se inițializează variabilele clasei *ObserverSelector*. Altfel se inițializează variabilele clasei *TargetSelector*.
9. Este setată dimensiunea câmpului *content*.
10. **Funcția TextSetup:**

public void TextSetup(TextMeshProUGUI txtObj, string text)

{

1. txtObj.rectTransform.sizeDelta = new Vector2(180, 30);
2. txtObj.text = text;
3. txtObj.color = Color.black;

txtObj.alignment = TextAlignmentOptions.Left;

txtObj.alignment = TextAlignmentOptions.Midline;

txtObj.fontSize = 30;

}

1. Este setată marimea câmpului de text.
2. Se atribuie textul.
3. Se efectuează setarile specifice textului: culoare, alinierea orizontală si verticală, mărimea fontului.
4. **Funcția ImageSetup:**

public void ImageSetup(Image img, int ord)

{

1. img.type = Image.Type.Sliced;
2. img.sprite = sprite;
3. img.rectTransform.sizeDelta = new Vector2(180, 30);
4. img.rectTransform.localPosition=new Vector3(91.5f, 15\*(celestialObjects.Count-26)-ord\*30,0);

}

1. Se setează tipul imaginii.
2. Se setează grafica obiectului.
3. Este setată mărimea campului imagine.
4. Este setată poziția sa în funcție de variabila *ord* si numarul de corpuri cerești.

ObserverSelector Class

**Utilizare:**

* Permite selecatrea corpului cerestru de pe care se fac observațiile.
* Este atașată unui obiect de tip *Button*.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using UnityEngine.UI;

public class ObserverSelector : MonoBehaviour

1. **Variabilele clasei:**
2. public SwitchCamera switchCamera;
3. List<CelestialObject> celestialObjects;
4. public int ordIndex;
5. Referință către un obiect de clasă *SwitchCamera*.
6. Lista corpurilor cerești.
7. Indexul corpului ceresc specific butonului.
8. **Funcția Start:**

public void Start()

{

celestialObjects = Initialize.GetCelestialObjects();

}

* Se inițializează lista de corpuri cerestre.

1. **Funcția OnClickEvent:**

public void OnClickEvent()

{

1. celestialObjects[switchCamera.cameraObserverIndex].trail.time = celestialObjects[switchCamera.cameraObserverIndex].trailTime;
2. celestialObjects[switchCamera.cameraObserverIndex].showTrail = true;
3. switchCamera.cameraObserverIndex = ordIndex;
4. switchCamera.observer = celestialObjects[ordIndex].gameObject;
5. switchCamera.obs.text = "Obs:" + switchCamera.observer.name;
6. celestialObjects[switchCamera.cameraObserverIndex].trail.time=0;
7. celestialObjects[switchCamera.cameraObserverIndex].showTrail = false;
8. switchCamera.UpdateScale(switchCamera.observer.transform.position.magnitude, celestialObjects[switchCamera.cameraTargetIndex].transform.position.magnitude);

switchCamera.Update();

}

1. Timpul orbitei observatorului precedent este reinițializat.
2. Afișarea orbitei este activată.
3. Se modifică indexul de cameră a observatorului.
4. Se schimbă corpul cerestru observator cu cel nou.
5. Se updatează textul.
6. Timpul de orbită a noului observator este setat la 0.
7. Afișarea orbitei este dezactivată.
8. Se apelează funcția *UpdateScale* și *Update* a obiectului *switchCamera*.

TargetSelector Class

**Utilizare:**

* Permite selectarea corpului cerestru *țintă*.
* Este atașată unui obiect de tip *Button*.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class TargetSeletor : MonoBehaviour

1. **Variabilele clasei:**
2. public SwitchCamera switchCamera;
3. List<CelestialObject> celestialObjects;
4. public int ordIndex;
5. Referință către un obiect de clasă *SwitchCamera*.
6. Lista corpurilor cerești.
7. Indexul corpului ceresc specific butonului.
8. **Funcția Start:**

public void Start()

{

celestialObjects = Initialize.GetCelestialObjects();

}

* Se inițializează lista de corpuri cerești.

1. **Funcția OnClickEvent:**

public void OnClickEvent()

{

1. switchCamera.cameraTargetIndex = ordIndex;

switchCamera.target = celestialObjects[ordIndex].gameObject;

1. switchCamera.targ.text = "Targ:" + switchCamera.target.name;
2. switchCamera.UpdateScale(switchCamera.target.transform.position.magnitude, celestialObjects[switchCamera.cameraObserverIndex].transform.position.magnitude);

switchCamera.Update();

}

1. Se schimbă indexul de cameră a țintei și corpul cerestru țintă.
2. Se updatează textul.
3. Se apelează funcția *UpdateScale* și *Update* a obiectului *switchCamera*.

**Scena**

**SolarSystem**

Initialize Class

**Utilizare:**

* Face legatura dintre *Settings* și *SolarSystem*.
* Conține variabilele specifice simulării.
* Creează lista corpurilor cerești participante la simulare.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Initialize : MonoBehaviour

1. **Variabilele clasei:**
2. public static float UA = 1.495978707e11f;
3. static int scale=10;
4. static bool showMoons=true;
5. static bool showOrbit=true;
6. static int timeMultiplier=12\*3600\*24;
7. public AnimationCurve volumeCurveInput;
8. static AnimationCurve volumeCurve;
9. static List<CelestialObject> celestialObjects;
10. static int maxCameraIndex;
11. Valoarea unitații astronomice(**UA**) in **m** (1**UA**≈1.495978707·1011m).
12. Valoarea scalei(default:**10** setată în meniul *Settings*).
13. Prezența sateliților naturali în simulare(default:**true**).
14. Orbita este sau nu vizibilă(default:**true**).
15. Viteza simulării(default:**12·3600·24s** sau **12zile** per *secundă*).
16. Funcția curbă([*AnimationCurve*](https://docs.unity3d.com/Manual/animeditor-AnimationCurves.html)) *input* a volumului(permite modificare).
17. Funcția curbă de tip **static** a volumului(permite accesarea în interiorul oricarei clase).
18. Lista corpurilor cerești.
19. Valoarea maximă a indexului de cameră.
20. **Funcția Start:**

public void Start()

{

1. volumeCurve = volumeCurveInput;
2. SetCelestialObjects();

}

1. Funcția curbă de tip static primește funcția curbă *input*.
2. Se apelează funcția de setare a corpurilor cerești.
3. **Funcțiile *Set* și *Get* ale variabilelor:**

public static void SetScale(int scaleInput)

{ scale = scaleInput; }

public static int GetScale()

{ return scale; }

public static void SetShowMoonsValue(bool showMoonsInput)

{ showMoons=showMoonsInput; }

public static bool GetShowMoonsValue()

{ return showMoons; }

public static void SetShowOrbitValue(bool showOrbitInput)

{ showOrbit = showOrbitInput; }

public static bool GetShowOrbitValue()

{ return showOrbit; }

public static void SetTimeMultiplier(int timeMultiplierInput)

{ timeMultiplier = timeMultiplierInput; }

public static int GetTimeMultiplier()

{ return timeMultiplier; }

static void SetMaxCameraIndex(int max)

{ maxCameraIndex = max; }

public static int GetMaxCameraIndex()

{ return maxCameraIndex; }

public static List<CelestialObject> GetCelestialObjects()

{ return celestialObjects; }

* Permit accesarea externă(cu excepție funcția SetMaxCameraIndex).

1. **Funcția SetCelestialObjects:**

public static void SetCelestialObjects()

{

1. celestialObjects = new List<CelestialObject>();
2. celestialObjects.AddRange(FindObjectsOfType<CelestialObject>());
3. celestialObjects.Sort(CompareByDistanceFromSun);
4. if (!showMoons)

for (int i = 0; i < celestialObjects.Count; i++)

{

if (celestialObjects[i].objectType==

CelestialObject.CelestialObjectType.Moon)

{

celestialObjects[i].gameObject.SetActive(false);

celestialObjects.RemoveAt(i);

i--;

}

}

1. SetMaxCameraIndex(celestialObjects.Count - 1);
2. ScaleCelestialObjects();

}

1. Se inițializează lista corpurilor ceresti.
2. Se adaugă in listă toate obiectele de tip *CelestialObjects*.
3. Lista este sortată in funcție de *CompareByDistanceFromSun*:

static int CompareByDistanceFromSun(CelestialObject a, CelestialObject b)

{

return a.numberOfOrder.CompareTo(b.numberOfOrder);

}

* *numberOfOrder*-al câtelea corp ceresc este în funcție de distanța până la Soare.
* Funcția *CompareTo* returnează: , astfel corpurile cerești pot fi sortate.

1. În cazul în care nu se dorește includerea sateliților in simulare aceștia sunt eliminați din listă și devin inactivi în simulare.
2. Se setează indexul maxim de cameră.
3. Se apelează funcția de calcul a mărimii corpului ceresc:

static void ScaleCelestialObjects()

{

1. float maxVolum = float.MinValue;

for (int i = 0; i < celestialObjects.Count; i++)

{

if (celestialObjects[i].volum > maxVolum)

maxVolum = celestialObjects[i].volum;

}

for (int i = 0; i < celestialObjects.Count; i++)

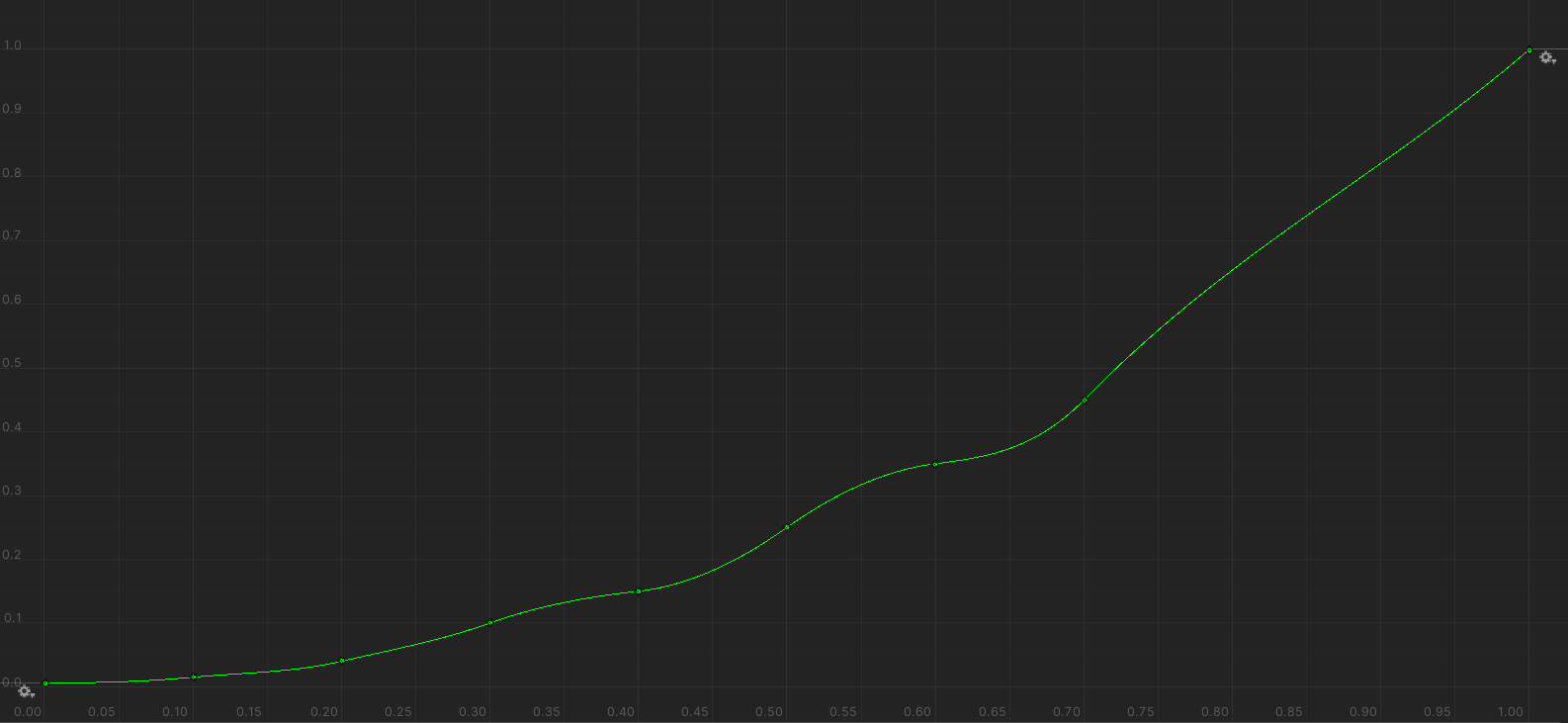
{

1. celestialObjects[i].scale = volumeCurve.Evaluate(celestialObjects[i].volum <= 1e16f ? 0 : Mathf.Log10(celestialObjects[i].volum / 1e16f) / Mathf.Log10(maxVolum / 1e16f));
2. celestialObjects[i].gameObject.transform.localScale = celestialObjects[i].scale \* 3 \* Vector3.one;

}

}

1. Se află volumul maxim.
2. Marimea corpului ceresc este calculată astfel:

* -volumul corpului (),
* -numărul de corpuri cerești-1
* 𝕍-mulțimea formată din valorile
* -funcție curbă arbitrară:
* -dimensiunea corpului ceresc

1. Se setează dimensiunea corpului în spatiu(marită de 3 ori pentru vizibilitate)

Gravity Class

**Utilizare:**

* Această clasă este folosită pentru a calcula forța de atracție gravitațională dintre corpurile cerești.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class Gravity : MonoBehaviour

1. **Variabilele clasei:**
2. List<CelestialObject> celestialObjects;
3. const float G = 6.67428e-11f;
4. Lista corpurilor ceresti.
5. Constanta gravitatională(G=6.67428·10-11**m3·kg-1·s-2**).
6. **Funcția Start:**

public void Start()

{

celestialObjects = Initialize.GetCelestialObjects();

}

* Se preia lista corpurilor cerești din clasa *Initialize*.

1. **Funcția FixedUpdate:**

public void FixedUpdate()

{

for(int i=0;i<celestialObjects.Count;i++)

{

1. Vector3 Fg = new Vector3(0, 0, 0);

for (int j=0;j<celestialObjects.Count;j++)

{

1. if (i!=j)

{

1. Vector3 R = (celestialObjects[j].transform.position-celestialObjects[i].transform.position) /

Initialize.GetScale() \* Initialize.UA;

1. float r = R.sqrMagnitude;
2. Fg += G \* celestialObjects[i].mass \* celestialObjects[j].mass / r \* R.normalized;

}

}

1. celestialObjects[i].a = Fg / celestialObjects[i].mass;

}

}

1. Pentru fiecare corp ceresc se inițializeazș *forța gravitaționalș* cu vectorul și se începe algoritmul de calcul a interacțiunii gravitaționale dintre corpurile cerești.
2. Se evită calculul forței gravitaționale cu sine însuși.
3. Se calculează vectorul de poziție dintre cele două corpuri după formula:

* -vectorul de poziție
* -vectorul de poziție a corpului în sistem
* -valoarea scalei
* -unitate astronomică

1. Se calculează patratul modulului vectorului de poziție
2. Se calculează forța gravitațională rezultantă:

* -forța gravitațională care acționează asupra corpului
* -numarul de corpuri cerești-1
* -constanta gravitațională
* -masa corpului
* -versorul lui ( este *normalizat* => )

1. Se calculează accelerația instantanee a corpului ceresc:

* -accelerația instantanee la momentul

CelestialObject Class

**Utilizare:**

* Această clasă este atribuită tuturor corpurilor cerestre.
* Conține toți parametri specifici corpurilor cerești din simulare.
* Calculul ecuației mișcării.

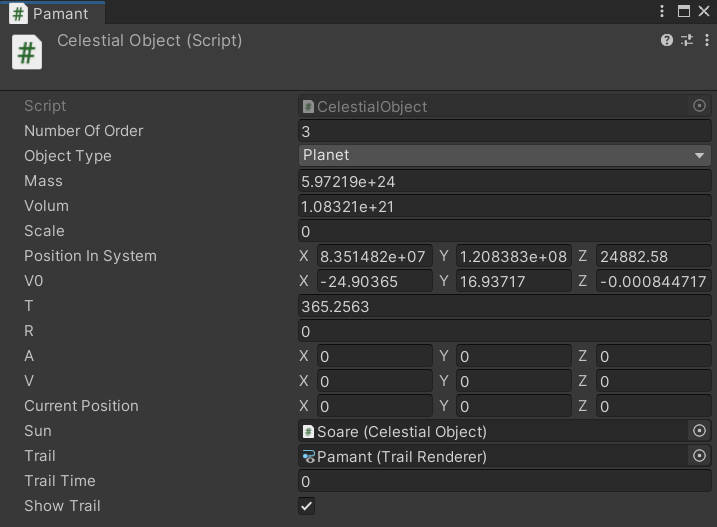
**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class CelestialObject : MonoBehaviour

1. **Variabilele clasei:**
   1. public int numberOfOrder;
   2. public enum CelestialObjectType { Sun, Planet, Moon };
   3. public CelestialObjectType objectType;
   4. public float mass;
   5. public float volum;
   6. public float scale;
   7. public Vector3 positionInSystem;
   8. public Vector3 v0;
   9. public float T;
   10. public Vector3 a;
   11. public Vector3 v;
   12. public Vector3 currentPosition;
   13. public CelestialObject Sun;
   14. public TrailRenderer trail;
   15. public float trailTime;
   16. public bool showTrail = true;
2. Al câtelea corp ceresc este de la Soare(pentru Soare numberOfOrder=0). Este folosit pentru a adăuga corpurile cerești într-o listă ordonată.
3. Lista de tipuri de corpuri cerești(Soare, Planeta, Luna).
4. Tipul specific corpului ceresc actual.
5. Masa corpului ceresc în **kg**.
6. Volumul în **m³**.
7. Mărimea corpului ceresc(diametrul). Este calculat în funcție de volumul corpurilor cerești în clasa *Initialize*.
8. Coordonatele inițiale (*x0,y0,z0*) ale corpului ceresc în **km**.
9. Viteza inițială (*vx0,vy0,vz0*) a corpului ceresc în **km·s-1**.
10. Perioada orbitală a corpului ceresc în **zile**.
11. Accelerația instantanee (*ax,ay,az*) a corpului ceresc în **m·s-2**.
12. Viteza instantanee (*vx,vy,vz*) a corpului ceresc în **m·s-1**.
13. Coordonatele curente (*x,y,z*) ale corpului ceresc în **m**.
14. Referință la corpul ceresc *Soare*.
15. Orbita corpului ceresc(se folosește componenta [Trail Renderer](https://docs.unity3d.com/Manual/class-TrailRenderer.html) din Unity).
16. Perioada orbitală relativă la durata unei zile în cadrul simularii.
17. showTrail = true/false(orbita este/nu este vizibilă)

*Datele inițiale ale planetei* ***Pământ*** *introduse*

*în fereastra* ***Inspector*** *din Unity*

1. **Funcția Start:**

private void Start()

{

1. trail.gameObject.SetActive(false);
2. positionInSystem\*=1000/Initialize.UA\*Initialize.GetScale();

transform.position = positionInSystem;

1. v0 \*= 1000;

v = v0;

1. trailTime = T \* 3600 \* 24/Initialize.GetTimeMultiplier();

trail.time = trailTime;

1. switch (objectType)

{

case CelestialObjectType.Sun:

trail.startWidth = 0.1f;

break;

case CelestialObjectType.Planet:

trail.startWidth = 0.1f;

break;

case CelestialObjectType.Moon:

trail.startWidth = 0.01f;

break;

}

1. trail.material= gameObject.GetComponent<Renderer>().material;
2. trail.gameObject.SetActive(Initialize.GetShowOrbitValue());

}

1. Orbita este setată *inactivă* pentru a seta poziția inițială a corpului cerestru.
2. Se face conversia din **km** în **m**, apoi în unități astronomice, apoi în unități de scală(1UA=1Scale). Se atribuie poziția calculată corpului ceresc.
3. Se face conversia din **km·s-1** în **m·s-1** și se atribuie vitezei instantanee.
4. Se calculează câte secunde durează o perioadă orbitală în condițiile simulării și se atribuie timpului componentei *trail*.
5. În funcție de tipul corpului cerestru se setează lațimea curbei orbitale.
6. Se atribuie orbitei aceeași culoare ca a corpului cerestru respectiv.
7. Orbita este setată în funcție de valoarea *showOrbit* din meniul *Settings*.
8. **Funcția FixedUpdate:**

private void FixedUpdate()

{

1. v+=a\*Time.fixedDeltaTime\*Initialize.GetTimeMultiplier();
2. transform.position+=Time.fixedDeltaTime\* Initialize.GetTimeMultiplier()\*v\*Initialize.GetScale()/

Initialize.UA;

1. currentPosition=transform.position/Initialize.GetScale()\* Initialize.UA;

}

1. Viteza este calculată după formula:

* -viteza instantanee la momentul
* -timeMultiplier(setată în meniul *Settings*)
* -fixedDeltaTime(durata de timp până la urmatorul [*FixedUpdate*](https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html) =0.01s)
* -accelerația instantanee a corpului cerestru la momentul

1. Poziția corpului este calculată după formula:

* -vectorul de poziție a corpului în sistem la momentul
* -valoarea scalei
* -unitate astronomica

1. Conversia coordonatelor corpului din **US** în **m**.

DayCounter Class

**Utilizare:**

* Afisează numarul de zile care au trecut de la începerea simularii și permite monitorizarea acesteia în timp.
* Ziua 0 reprezintă data de 18.11.2023.

**Definiție:**

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using TMPro;

using UnityEngine.UI;

public class DayCounter : MonoBehaviour

1. **Variabilele clasei:**
2. public float days = 0;
3. public TMP\_Text daysCounter;
4. Contorul care numară zilele.
5. Referință la un obiect de tip *text*.
6. **Funcția FixedUpdate:**

public void FixedUpdate()

{

1. days += Time.fixedDeltaTime \* Initialize.GetTimeMultiplier() / (3600 \* 24);
2. daysCounter.text = "Day:" + ((int)days).ToString();

}

1. Se incrementează numărul de zile cu: .
2. Textul se rescrie înlocuind numărul anterior de zile cu numărul nou calculat.

**HotKeys**

* Fullscreen-F1
* Back-Escape
* SelectMenu-Space
* Next Observer-RightArrow
* Previous Observer-LeftArrow
* Next Target-UpArrow
* Previous Target-DownArrow

**Videoclipuri/**

**Poze**

* [Sistemul Solar Interior](Foto_Video/Sistemul%20Solar%20Interior.mp4)
* [Jupiter observat de pe Callisto](Foto_Video/Jupiter%20observat%20de%20pe%20Callisto.mp4)
* [Saturn observat de pe Iapetus](Foto_Video/Saturn%20observat%20de%20pe%20Iapetus.mp4)
* [Uranus observat de pe Oberon](Foto_Video/Uranus%20observat%20de%20pe%20Oberon.mp4)
* [Blue Dot](Foto_Video/Blue%20Dot.png)