Proiect

Grafica Digitala

-Joc Unity-

Student: Lapuste Andrei Raul

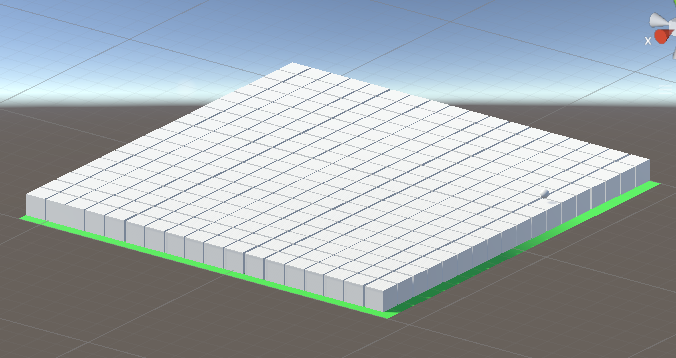
grupa 30143

1.Prezentarea proiectului

Folosind platforma Unity , un game-engine pentru dezvoltarea jocurilor am creat un joc simplu in care jucatorul trebuie sa traverseze traseul si sa ajunga la punctul de final. Obiectul controlat este o sfera simpla capabila de miscare pe axele x-z-y. Traseul are la baza o retea(matrice) de 324 cuburi. Odata cu pornirea jocului , butonul play din fereastra Unity, jocul porneste si rearanjeaza planul in 6 feluri , astfel avand la dispozitie 6 traseuri ce trebuie parcurse unul cate unul. Odata atins punctul de finish, cuburile se repozitioneaza in forma de baza. Odata atinsa forma de baza , urmatorul nivel ia forma si jucatorul se repozitioneaza la punctul de start(cub 125). Pentru a face traseul mai interesant , anumite cuburi au fost programate sa se miste pe una dintre cele 3 axe. Codul a fost scris in C#.

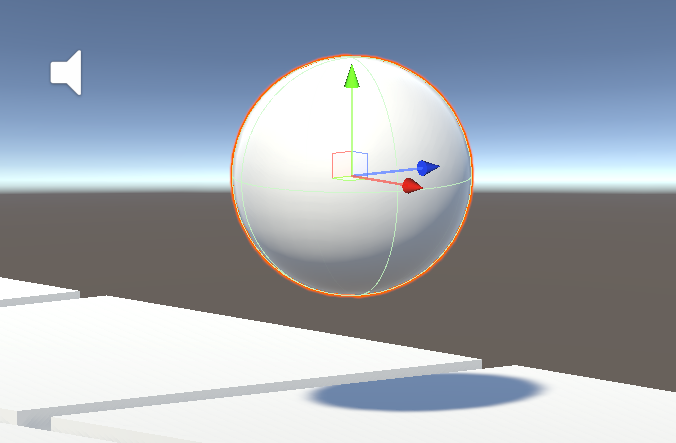
* 1. Prezentare plan de baza

Planul de baza este format din 324 de cuburi dispersate intr-un patrat , sustinute de un plan pentru a combate gravitatia si a nu cadea spre infiniti.



* 1. Jucatorul

Obiectul controlat este o sfera , capabila de deplasare pe axele x-y-z . Aceasta poate fi controlata folosind sagetile tastaturii sau tastele W,A,S si D , bara SPACE pentru a sari si tasta R pentru a reveni la start in cazul in care te-ai indepartat prea tare sau nu poti reveni pe traseu. La apasarea tastei SPACE se activeaza un sunet iar sfera nu poate sari iar pana cand atinge o suprafata.



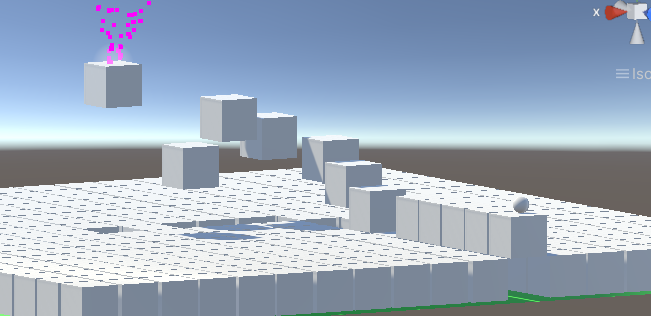
* 1. Punctul de finish

Fiecare nivel are un punct de finish diferit marcat ca umator:

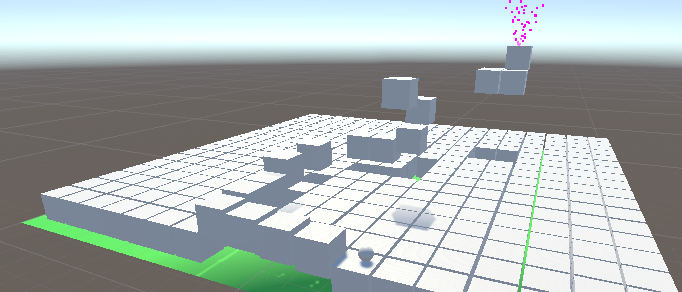


* 1. Prezentare nivele

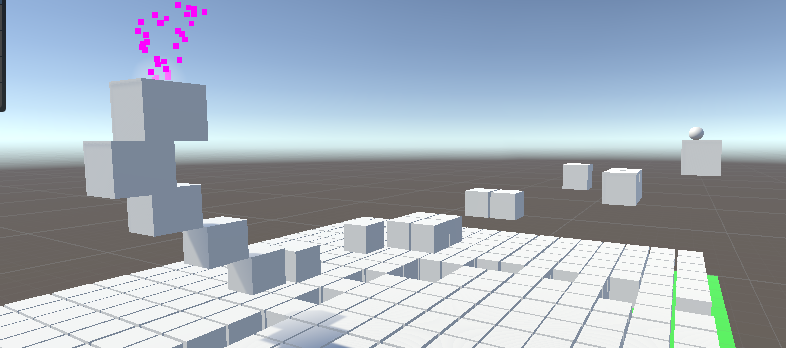
-nivelul 1



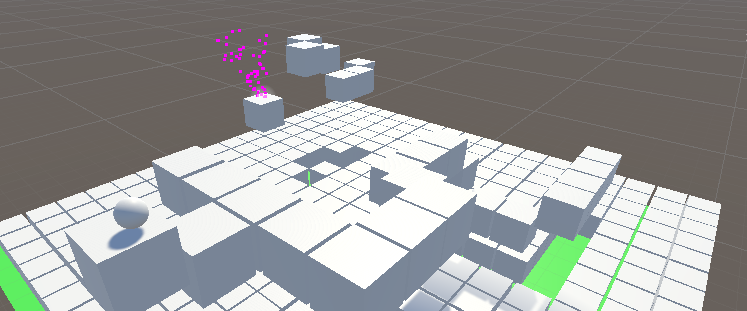
-nivelul 2



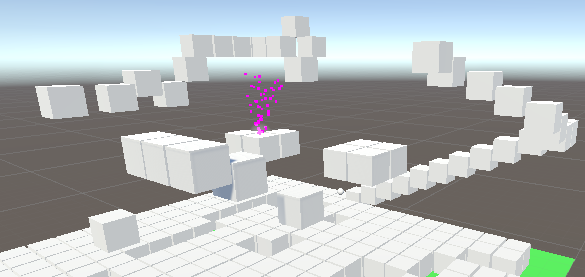
-nivelul 3



-nivelul 4



-nivelul 5



-nivelul 6



2.Explicarea codului

2.1 Main Camera

Secvena de cod face camera sa urmareasca bila, target, transmisa ca si obiect de tip Trasnform, dintr-o pozitie fixata “diff” de tip Vector3.

public class CameraFollow : MonoBehaviour

{

// Start is called before the first frame update

void Start()

{

}

public Transform target;

public Vector3 diff;

// Update is called once per frame

void Update()

{

transform.position = target.position + diff;

transform.LookAt(target.position);

}

}

2.2 Matricea de baza

Cuburile nu au propriul lor cod , acestea au totusi adaugate componente 

Componenta Rigidbody are optiunea is Kinematic bifata, astfel acestea nu se pot misca decat prin utilizarea procedurilor “transform”. Mesh Collider permite face cuburile impermeabile cand jucatorul atinge suprafata.

2.3 Obiectul sfera



Proprietatile fizice ale sferei sunt caracterizate si urmarite de componenta Rigidbody, Mass(5) , Drag(5) , Angular Drag(0.05) si este supusa gravitatie. Scriptul Ball Controll contine codul aferent sferei. Acesta are ca parametri transmisi viteza x-z a sferei(Speed) , viteza de salt(Jump Speed) , rigid(componenta Rigidbody a sferei), pozitia de start(Resetcube) , o sursa pentru sunetul de salt(AudioSource) si BallJump (Vector3 pentru definirea distantei de salt a sferei in functie de JumpSpeed).

public class BallControll : MonoBehaviour

{

public float speed;

public float jumpSpeed ;

public Rigidbody rigid;

private bool jjp = true;

public GameObject resetcube;

public AudioSource src;

public Vector3 balljump = new Vector3(0f, 5.0f, 0f);

// Start is called before the first frame update

void Start()

{

rigid = gameObject.GetComponent<Rigidbody>();

}

// Update is called once per frame

void Update()

{

recetPosition();

movement();

if (Input.GetKey(KeyCode.Space) && jjp == true)

{

src.Play();

rigid.AddForce(balljump \* jumpSpeed);

jjp = false;

}

if (rigid.velocity.y < 0 )

{

rigid.AddForce(balljump \* -jumpSpeed/47);

}

}

private void OnCollisionStay()

{

if (rigid.velocity.y > -20)

jjp = true;

}

public void movement()

{

if (Input.GetAxis("Vertical") > 0)

{

rigid.AddForce(-Vector3.forward \* speed);

}

if (Input.GetAxis("Vertical") < 0)

{

rigid.AddForce(Vector3.forward \* speed);

}

if (Input.GetAxis("Horizontal") < 0)

{

rigid.AddForce(Vector3.right \* speed);

}

if (Input.GetAxis("Horizontal") > 0)

{

rigid.AddForce(Vector3.left \* speed);

}

}

public void recetPosition()

{

if (Input.GetKeyDown(KeyCode.R))

{ transform.position = resetcube.transform.position + new Vector3(0, 20, 0); }

}

public void finishRecet()

{

transform.position = resetcube.transform.position + new Vector3(0, 20, 0);

}

}

Functia recetPosition() si movement() sunt apelate constant datorita functiei Update(). Acestea asteapta validarea imputurilor. Functia recetPosition() asteapta apasarea tastei “R” iar functia movement() inputuri de la sageti sau W,A,S sau D.

Fucntia finishRecet() este preluata eventual de GameObjectul FinishPoint pentru a repozitiona sfera la punctul de start.

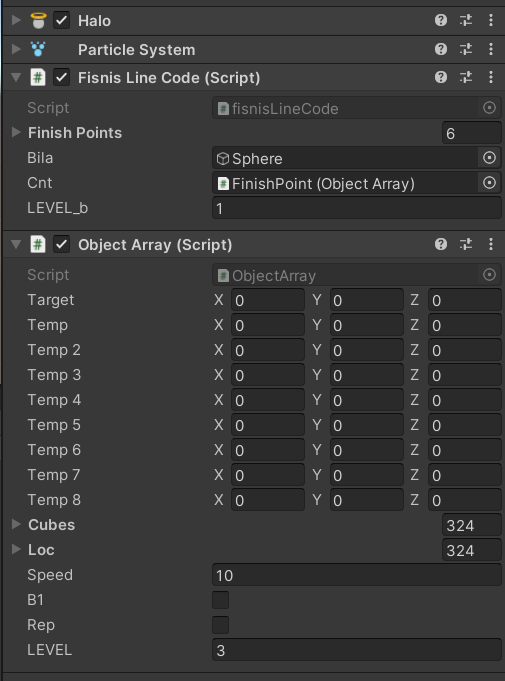
Secventa “if (Input.GetKey(KeyCode.Space) && jjp == true)” asteapta apasarea tastei SPACE si ca suntem in contact cu o suprafata

Conditia if (rigid.velocity.y < 0 ) urmareste sfera in aer si daca conditia este adevarata, face bila sa cada mai repede, ajutand fluxul jocului si neavand nevoie sa schimbat valoarea gravitatiei.

OnCollisionStay() se apeleaza cand sfera se loveste de de o suprafata(jjp devine true). Daca viteza de cadere a sferei este mai mare ca 20( velocity < -20 in jos) nu permite sferei sa sara. Aceasta actiune ofera posibilitatea unui double jump.

2.4 Obiectul FinishPoint

Pentru a iesi in evidenta , obiectul FinishPoint ,de tip GameObject, i-a fost adaugat evectele “Halo” si “Particle System” .



Scriptul “Object Array” este folosit pentru manipularea cuburilor , pozitionarea acestora pentru fiecare nivel si generarea animatilor. Parametrul target e folosit in transformarea pozitie cuburilor. Temp\_i memoreaza pozitile anumitor cuburi prestabilite odata ce traseul a fost construit. Aceste cuburi vor executa anumite deplasari pe x-y-z . Variabila Cubes primeste toate cuburile existente. Loc este un vector de tip Vector3[ ] care memoreaza forma de baza a matricei , Speed este viteza de deplasare a cuburilor , LEVEL prezinta nivelul( traseul ) si valorile B1 si Rep sunt variabile booleene folosite pentru repozitionari/ pozitionari si animatii.

Nivelurile sunt codate dupa 6 matrici cu 18 randuri, respectiv 18 coloane, exemplu matricea pentru nivelul 1;

double[,] l1 = {

{ 0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 } ,

{ 0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,0 ,25 ,30 ,35 ,0 ,40 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,20 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,15 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,10 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,10 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,10 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,10 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,10 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },

{ 0 ,0 ,0 ,0 ,0 ,0 ,10 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 ,0 },};

Functia void get\_initial\_position() este apelata in start() si memoreaza in “loc” pozitia fiecarui cub.

Functia void Update( ) contine un switch a carui cazuri sunt selectate dupa valoarea parametrului LEVEL. Cazurile 1->6 apeleaza functia fuynny() ce primeste ca parametru matricea nivelului care modeleaza traseul daca b1 and rep sunt “true” . Odata cuburile pozitionate dupa matricea nivelului , b1 si rep sunt trecute “false” folosind functia verificare(). Odata pozitionate si rep si b1 “false” ,animatile se activeaza.

Functile leftRight(), UpDown() si forwBack() sunt folosite pentru deplasarea anumitor cuburi date ca parametru fata de un punct ales in plan. Alti parmatetri reprezinta diferenta admisa dintre pozitile celor doua puncta.

public void leftRight(int ee , int max ,int min,int gg)

{

if (b1 ==false)

{

if (cubes[ee].transform.position.z - temp.z > max)

k[gg] = -1;

if (cubes[ee].transform.position.z - temp.z < min)

k[gg] = 1;

cubes[ee].transform.position += k[gg] \* Vector3.forward \* Time.deltaTime \* speed;

}

}

public void UpDown(int ee, int max, int min,int gg)

{

if(b1 ==false)

{

if (cubes[ee].transform.position.y - temp.y >=max)

k[gg] = -1;

if (cubes[ee].transform.position.y - temp.y <=min)

k[gg] = 1;

cubes[ee].transform.position += k[gg] \* Vector3.up \* Time.deltaTime \* speed;

}

}

public void forwBack(int ee, int max, int min,int gg)

{

if (b1 == false)

{

if (cubes[ee].transform.position.x - temp.x > max)

k[gg] = -1;

if (cubes[ee].transform.position.x - temp.x < min)

k[gg] = 1;

cubes[ee].transform.position += k[gg] \* Vector3.right \* Time.deltaTime \* speed;

}

}

Scriptul “FinishLineCode” se foloseste de obiectul sfera si scriptul Object Array pentru manipularea jocului. Acest script serveste ca un fel de controller pentru joc. Primeste ca si parametri scriptul Object Array , o lista cu cuburile ce servesc ca si punct de finish si LEVEL\_b.

Functia Void Start() seteaza b1 si rep existenti in cnt (Object Array) la valoarea “true”

Conditia if(Vector3.Distance(transform.position, bila.transform.position) <= 5) verifica daca sfera s-a apropiat destul de punctul de finish pentru a trece la umatorul nivel.