Introducere în Programarea Jocurilor pe Calculator

Cursul 5

Quick Maths - 2D

Coordonate 2D:

Vector2 direction = new **Vector2**(0.0f, 2.0f); // (x,y)
Vectori predefiniti:
var up   = Vector2.up;   // (0, 1)
var down = Vector2.down; // (0, -1)
var left = Vector2.left; // (-1, 0)
var right = Vector2.right; // (1, 0)
var one   = Vector2.one;   // (1, 1)
var zero = Vector2.zero; // (0, 0)

Quick Maths - 3D

Coordonate 3D:

Vector3 point= new **Vector3**(0.0f, 2.0f,3.5f); // (x,y,z)
Vectori predefiniti:

```
var up      = Vector3.up;      // ( 0,  1,  0)
var down    = Vector3.down;    // ( 0, -1,  0)
var left    = Vector3.left;    // (-1,  0,  0)
var right   = Vector3.right;   // ( 1,  0,  0)
var forward = Vector3.forward; // ( 0,  0,  1)
var back    = Vector3.back;    // ( 0,  0, -1)
var one     = Vector3.one;     // ( 1,  1,  1)
var zero    = Vector3.zero;    // ( 0,  0,  0)
```

Quick Maths

Detectarea orientarii unui obiect (orientarea axei locale Z):

Vector3 objectForward= transform.forward; //orientarea obiectului

Exemplul urmator arata cum putem deplasa un obiect inainte, in functie de orientarea lui curenta. (next slide)

Quick Maths - Exemplul 1:

```
public class Example : MonoBehaviour
{
   Rigidbody m_Rigidbody;
   void Start()


    {
        //Fetch the Rigidbody component you attach from your GameObject
        m_Rigidbody = GetComponent<Rigidbody>();
        //Set the speed of the GameObject
     }


   void Update()
    {
        if (Input.GetKey(KeyCode.UpArrow))
        {
            //Move the Rigidbody forwards constantly at speed you define (the blue arrow axis in Scene view)
            m_Rigidbody.velocity = transform.forward * (10.0f);
         }


 }
```

Quick Maths - Operatii de baza

```
var v1 = new Vector3(1f, 2f, 3f);
var v2 = new Vector3(0f, 1f, 6f);

var v3 = v1 + v2; // (1, 3, 9)
var v4 = v2 - v1; // (-1, -1, 3)
```

Quick Maths - Magnitude

Magnitudinea este “lungimea vectorului”. Altfel spus este radical din suma patratelor fiecarei componente a vectorului.

Exemplu:

var forwardMagnitude = Vector3.forward.magnitude; // = 1

var vectorMagnitude = new Vector3(2f, 5f, 3f).magnitude; // ~= 6.16

Quick Maths - Distanta

Daca magnitudinea lui (x,y,z) este distanta de la originea (0,0,0) la (x,y,z), cum aflam distanta dintre doi vectori?

Varianta 1:

```
var point1 = new Vector3(5f, 1f, 0f);
var point2 = new Vector3(7f, 0f, 2f);
var distance = (point2 - point1).magnitude; // = 3
```

Varianta 2:

```
var distance=Vector3.Distance(point1, point2);
```

Quick Maths - Observatie 1

Uneori nu avem nevoie de distanta ca si numar, ci doar sa vedem care pereche de obiecte sunt mai apropiate. In acest caz operatia de radical este inutila, si chiar costisitoare pentru a fi apelata la fiecare frame.

Astfel putem sa scriem:

```
var point1 = new Vector3(5f, 1f, 0f);
var point2 = new Vector3(7f, 0f, 2f);
var distanceSquared = (point2 - point1).sqrMagnitude; // = 9
```

Quick Maths - Observatie 2

De multe ori, spre exemplu in cazul orientarilor, este util sa folosim vectori de magnitudine 1. Altfel spus, odata ce avem un vector, este nevoie sa il normalizam.

var bigVector = new Vector3(4, 7, 9); // magnitude = 12.08
var unitVector = bigVector / bigVector.magnitude; // magnitude = 1
sau, mai simplu:
var unitVector2 = bigVector.normalized;

Quick Maths: Scalare

Inmultirea unui vector cu un numar

var v1 = Vector3.one * 4; // = (4, 4, 4)

Inmultirea a 2 vectori pe fiecare componenta (produs scalar):

v1.Scale(new Vector3(3f, 1f, 0f)); // = (12f, 4f, 0f)

Quick Maths: Produs scalar

Produsul scalar sau produsul Dot, este “diferenta” dintre directiile inspre care arata cei doi vectori. Formal spus, este suma produselor pe fiecare componenta. Consecinta: obtinem valoara cosinusului pentru unghiul dintre cei doi vectori.

- Produsul vectorial pentru 2 vectori normalizati, paraleli, de aceasi orientare este 1:
    - var parallel = Vector3.Dot(Vector3.left, Vector3.left);
- Produsul a doi vectori normalizati, paraleli, dar de orientari opuse este -1:
    - var opposite = Vector3.Dot(Vector3.left, Vector3.right); // -1
- var orthogonal =Vector3.Dot(Vector3.up, Vector3.forward); // 0

Quick Maths: Produs Vectorial

Produsul vectorial este o operatie binara ce are ca rezultat un nou vector perpendicular pe planul celor 2 vectori, si de magnitudine egala cu produsul magnitudinilor celor doi vectori inmultit cu sinusul unghiului dintre ei.

Orientarea se afla aplicand regula mainii **DREPTE**.

Exemplu:

var up = Vector3.Cross(Vector3.forward, Vector3.right);

Vector3: MoveTowards

public static Vector3 **MoveTowards**(Vector3 **current**, Vector3 **target**, float **maxDistanceDelta**);

Se returneaza vectorul pe a carei orientare trebuie sa translatam un punct de pe pozitia current catre pozitia target, fara a depasi maxDistanceDelta la fiecare pas (vectorul rezultat are magniturine maxDistanceDelta)

var moved =Vector3.MoveTowards(Vector3.zero, Vector3.one, 0.5f);

// = (0.3, 0.3, 0.3) (magnitudine ~0.5)

Vector3: Lerp

public static Vector3 **Lerp**(Vector3 **a**, Vector3 **b**, float **t**);

Interpoleaza vectorul **a** si vectorul **b** folosind parametrul **t**. Se retuneaza vectorul **(b-a)*t,** pentru **t=0**, se retuneaza **a**, pentru **t=1** se returneaza **b.**

var lerped = Vector3.Lerp(Vector3.zero, Vector3.one, 0.65f); // = (0.65, 0.65, 0.65)

Daca **t** nu se afla in [0,1] atunci el va fi normalizat. Pentru a folosi un **t** oarecare, se poate folosi LerpUnclamped.

var unclamped = Vector3.LerpUnclamped(Vector3.zero, Vector3.right, 2.0f);

// = (2, 0, 0)

Cum ne dam seama ca doua obiecte sunt apropiate unul de altul?

```
public class RangeChecker : MonoBehaviour {


    // The object we want to check the distance to
        [SerializeField] Transform target;
        [SerializeField] float range = 5;
        private bool targetWasInRange = false;
```

```
void Update () {
        var distance = (target.position - transform.position).magnitude;
        if (distance <= range && targetWasInRange == false) {
            Debug.LogFormat("Target {0} entered range!", target.name);
            targetWasInRange = true; }
else if (distance > range && targetWasInRange == true) {
                Debug.LogFormat("Target {0} exited range!", target.name);
            targetWasInRange = false;}}


}
```

Cum gasim unghiul la care trebuie rotit un obiect pentru a fi cu fata catre altul?

```
public class AngleChecker : MonoBehaviour {

 [SerializeField] Transform target;

   void Update () {

var directionToTarget = (target.position - transform.position).normalized;

var dotProduct = Vector3.Dot(transform.forward, directionToTarget);

var angle = Mathf.Acos(dotProduct);
Debug.LogFormat(
          "The angle between my forward direction and {0} is {1:F1}°",
          target.name, angle * Mathf.Rad2Deg
       );}}
```