Báo cáo mô phỏng hệ thống ném xiên

1. Giới thiệu
2. Cơ sở lý thuyết

* Xét vật được ném từ độ cao h0 (m), góc ném anpha (độ), vận tốc vo (m/s), ta phân tích thành 2 giai đoạn chuyển động ném xiên và ném ngang
* Với chuyển động ném xiên ta phân tích chuyển động theo trục x và trục y
  + Xét gia tốc:
    - ax = 0
    - ay = -g
  + Xét vận tốc:
    - vx = v0\*cos(anpha)
    - vy = v0\*sin(anpha) – gt;
  + Xét tọa độ
    - x = v0\*cos(anpha)\*t
    - y = v0\*sin(anpha)\*t – gt^2/2
* Với chuyển động ném ngang:
  + Xét gia tốc:
    - ax = 0
    - ay = -g
  + Xét vận tốc:
    - vx = v0
    - vy = -gt;
  + Xét tọa độ
    - x = v0\*t
    - y = - gt^2/2

Xử lý chuyển động:

transform.position = p + new Vector3(vo \* Mathf.Cos(anpha) \* t, vo \* Mathf.Sin(anpha) \* t - Setting.g \* t \* t / 2)

với:

* p là tọa độ ban đầu của vật
* Setting.g = 10
* thời gian t sẽ được cập nhật theo từng frame: t += Time.deltaTime;

Xử lý chiều cao tối đa:

* Tại thời điểm độ cao cực đại vật đang ở giai đoạn chuyển động ném xiên và vy = 0

if(vo \* Mathf.Sin(anpha) - Setting.g \* t <= 0.05f && !showH)

{ // xét sai số trong phạm vi 0.05f và biến boolean showH cho biết đã hiển thị chưa

showH = true;

points.Add(Instantiate(prePointSpawn, transform.position, Quaternion.identity)); // tạo điểm cực đại

points[points.Count - 1].transform.SetParent(goParent.transform);

points.Add(Instantiate(goCanva, transform.position, Quaternion.identity)); // Tạo thông tin hiển thị

//Debug.Log(y0);

float h = (transform.localPosition.y - pDat.y) \* Setting.tiLe; // độ cao h = vị trí hiện tại – vị trí gốc

string s = "Tầm cao: " + h.ToString("F2") + " m, tai t = " + t.ToString("F2") + " s";

points[points.Count - 1].transform.GetChild(0).GetComponent<TextMeshProUGUI>().text = "h = " + h.ToString("F2") + " m";

controller.UpdateTxt(s, 0); // cập nhật thông báo về chiều cao

points[points.Count - 1].transform.SetParent(goParent.transform); // thêm vào danh sách điểm

}

Xử lý tầm xa:

if(transform.position.y <= pDat.y && check) // khi vật chạm đất

{

check = false; // tắt cập nhật trạng thái vật

points.Add(Instantiate(goCanva, transform.position, Quaternion.identity));

float l = (transform.localPosition.x - x0) \* Setting.tiLe; // tầm xa = vị trí hiện tại – vị trí ban đầu

string s = "Tầm xa: " + l.ToString("F2") + " m, tai t = " + t.ToString("F2") + " s";

points[points.Count - 1].transform.GetChild(0).GetComponent<TextMeshProUGUI>().text = "l = " + l.ToString("F2") + " m";

controller.UpdateTxt(s, 1); // cập nhật thông báo về tầm xa

points[points.Count - 1].transform.SetParent(goParent.transform);

tunnel.Show(transform.localPosition);

}

Xử lý hiển thị quỹ đạo vật:

if(\_time < 0) // tạo điểm quỹ đạo sau mỗi time\_spawn (s)

{

maxPoints--; // giới hạn số lượng điểm

if(maxPoints > 0)

{

points.Add(Instantiate(prePointSpawn, transform.position, Quaternion.identity));

points[points.Count - 1].transform.SetParent(goParent.transform);

}

\_time = time\_spawn;

}

Xử lý thông tin hiển thị:

public void UpdateMuc() // hiển thị vị trí h

{

goMuc.transform.position = pMuc + new Vector3(0f, sldMuc.value\*maxHeight, 0f);

sldMuc.transform.GetChild(3).GetComponent<TextMeshProUGUI>().text = "Chiều cao: " + (pSpawnBullet.position.y - pMuc.y).ToString("F2") + " m";

}

public void UpdateGoc() // hiển thị vị trí về góc, đồng thời tính thêm độ cao khi xoay góc – vì điểm bắn với vị trí đối chiếu khác nhau

{

goNongSung.transform.rotation = Quaternion.Euler(0f, 0f, sldNongSung.value);

sldNongSung.transform.GetChild(3).GetComponent<TextMeshProUGUI>().text = "Góc bắn: " + (sldNongSung.value).ToString("F2") + " độ";

sldMuc.transform.GetChild(3).GetComponent<TextMeshProUGUI>().text = "Chiều cao: " + (pSpawnBullet.position.y - pMuc.y).ToString("F2") + " m";

}

public void UpdateLuc() // hiển thị vận tốc

{

veclocity = sldLuc.value \* maxVeclocity;

sldLuc.transform.GetChild(3).GetComponent<TextMeshProUGUI>().text = "Tốc độ: " + (veclocity).ToString("F2") + " m/s";

}

public void UpdateTxt(string s, int type) // hiển thị thông tin về tầm cao (type = 0), tầm xa (type khác 0)

{

if(type == 0)

txtH.text = s;

else

txtL.text = s;

}

Hiển thị kết quả đạt được

public class Tunnel : MonoBehaviour

{

[SerializeField] float[] distances = { 0.5f, 1f, 1.5f }; // phạm vi điểm

// Start is called before the first frame update

[SerializeField] GameObject txtScore;

public void Show(Vector3 p)

{

float distance = Vector2.Distance(transform.position, p);

if(distance < distances[0]) // nằm trong phạm vi 0

{

SettingTxt("A");

}else if(distance < distances[1]) // nằm trong phạm vi 1

{

SettingTxt("B");

}else if (distance < distances[2]) // nằm trong phạm vi 2

{

SettingTxt("C");

}else SettingTxt("F"); // TH còn lại

}

void SettingTxt(string s)

{

GameObject gameObject = Instantiate(txtScore, transform.position, Quaternion.identity);

gameObject.transform.GetChild(0).GetComponent<TextMeshProUGUI>().text = s;

Destroy(gameObject, 1f); // hủy đối tượng sau 1s

}

}

Xử lý tạo và hủy

public void Spawn()

{

\_number--;

if(\_number == 0) // khi số lượng đối tượng mô phỏng hết

{

\_number = number;

for (int i = goSpawn.transform.childCount - 1; i >= 0; i--) // hủy các đối tượng vừa tạo

{

// Lấy đối tượng con thứ i

GameObject childObject = goSpawn.transform.GetChild(i).gameObject;

// Hủy bỏ đối tượng con

Destroy(childObject);

}

}

// tạo đối tượng mới đồng thời cài đăt các chỉ số tương ứng

GameObject gameObject = Instantiate(preBullet, pSpawnBullet.position, Quaternion.identity);

gameObject.transform.SetParent(goSpawn.transform);

BulletController bulletController = gameObject.GetComponent<BulletController>();

bulletController.anpha = sldNongSung.value \* (Mathf.PI / 180f);

bulletController.vo = veclocity;

bulletController.goParent = goSpawn;

bulletController.pDat = pMuc;

bulletController.x0 = pSpawnBullet.position.x;

bulletController.check = true;

bulletController.controller = this;

bulletController.tunnel = goTunnel.transform.GetComponent<Tunnel>();

txtH.text = "Tầm cao tối đa: ... m" ;

txtL.text = "Tầm dài tối đa: ... m" ;

}

Tạo đích bắn

public void NewEnemy()

{

if (goTunnel != null) // hủy đối tượng cũ

Destroy(goTunnel);

goTunnel = Instantiate(preTunnel, new Vector3(Random.Range(pMuc.x + 2, maxX), pMuc.y + 0.3f, 0), Quaternion.identity); // tạo đối tượng mới trong phạm vi màn hình

txtD.text = "Khoảng cách: " + (goTunnel.transform.position.x - pMuc.x).ToString("F2") + " m"; // hiển thị phạm vi

}

Ảnh có chứa văn bản, ảnh chụp màn hình, núi

Mô tả được tạo tự động

Màn hình thao tác

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

Thanh slider cho phép điều chỉnh thông số ban đầu

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu tượng, Phông chữ

Mô tả được tạo tự động

Nút xử lý bắn, thoát ứng dụng, tạo bia mới

Ảnh có chứa văn bản, Phông chữ, màu trắng, thiết kế

Mô tả được tạo tự động

Vùng hiển thị thông tin trực quan

Ảnh có chứa bầu trời, Tác phẩm nghệ thuật của trẻ con, núi, ảnh chụp màn hình

Mô tả được tạo tự động

Màn hình hiển thị

1. Kết quả đạt được
2. Kết luận

Ưu điểm:

* Mô phỏng một cách xác thực về hiện tượng vật lý
* Thao tác dễ dàng, ưa nhìn
* Các thông tin cần thiết rõ rang

Hạn chế:

* Vật thể còn cố định về trục X
* Giới hạn phạm vi giá trị chưa được lớn

Hướng phát triển:

* Mô phỏng thêm các tác nhân như lực cản, thay đổi gia tốc g, …
* Thay đổi vị trí vật trong không gian 2 chiều
* Hiển thị các vector vận tốc khi chuyển động