

Programación Visual

Presentación del trabajo final de la asignatura

Daniel Ramírez Brescia.

¿Cómo se juega?.



La parte fundamental del juego es como se mueve el personaje , como choca con la caja y como esta se mueve.



Elementos principales del juego

- Movimiento.
- Choques.
- Función ganar juego.
- Vista global

Implementación

```
public class Game1 : Microsoft.Xna.Framework.Game
{
    Atributos generales
    Métodos de inicializacion y carga

    protected override void Update(GameTime gameTime)...

    protected override void Draw(GameTime gameTime)...
    Métodos de dibujo
}
public class personaje
{
    Parámetros
    Constructor de clase
    Funciones auxiliares de Movimiento Personaje.
    Movimientopersonaje
    Escena
    choques
}
public class CajasMoviles
{
    Parámetros
    Constructor de clase
    MovimientoCajas
    funcion axuliar de movimiento caja
    Escena

}
```

Movimiento Personaje

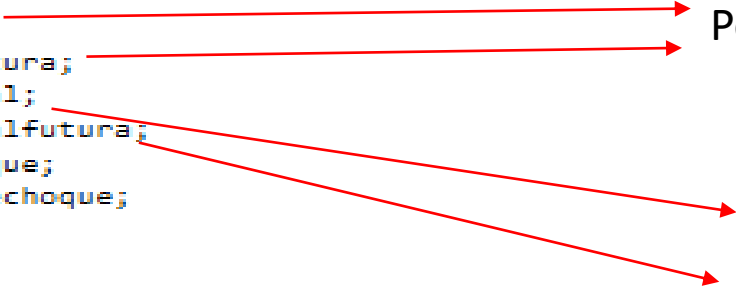


- El jugador se controla mediante las teclas w , a ,s d.
- Una vez inicializado en movimiento en una dirección no puede cambiarla.
- Puede continuar moviéndose en la misma dirección si se mantiene pulsada la tecla.
- Todo el código que realiza lo descrito se encuentra en la clase personaje.

Clase personaje

Como hemos visto en clase para la realización del movimiento se ha utilizado dos variables que almacenan posición en la matriz y otra que almacena la posición en el mundo virtual.

```
#region Parámetros
public Vector3 posicion;
public Vector3 posicionfutura;
public Vector3 posicionreal;
public Vector3 posicionrealfutura;
public Vector3 puntodechoque;
public Vector3 direcciondechoque;
Vector3 desplaza;
int ide;
float distanciainicial = 0;
float distanciaactual = 0;
float boxmesure;
public float speed;
    bool trans=false;
public bool choque = false;
    int actualizado;
    float starttime;
    float currenttime;
```



Posición en la matriz

Posición real en el mundo

The diagram consists of red arrows pointing from specific lines of code to text labels on the right. One arrow points from 'posicion;' to 'Posición en la matriz'. Another arrow points from 'posicionreal;' to 'Posición real en el mundo'. A third arrow points from 'posicionrealfutura;' to 'Posición real en el mundo'. A fourth arrow points from 'puntodechoque;' to 'Posición real en el mundo'.

El programa.

```
protected override void Update(GameTime gameTime)
{
    if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
        this.Exit();

    ganarjuego();
    tiempo = (float)gameTime.TotalGameTime.TotalSeconds;

    Jugador.MovimientoPersonaje(tiempo,map);
    if (Jugador.choque)
        caja.Movimientocajas(ref Jugador.direcciondechoque, ref Jugador.puntodechoque, ref map, ref Jugador.choque);

    if (Keyboard.GetState().IsKeyDown(Keys.V))
    {
        cameraPosition = Escena(8, 8, 0);
        cameraCentrado = Escena(8, 8, 0);
        cameraPosition.Z += 2000.0f;
    }
    else
    {
        cameraPosition = Jugador.posicionreal;
        cameraCentrado = Jugador.posicionreal;
        cameraPosition.Z += 700.0f;
    }

    base.Update(gameTime);
}
```

Partes a explicar.


```

public void MovimientoPersonaje(float tiempo , int [,] mapa)
{
    if (trans==false)
    {
        actualizado = botonpulsado();
        actualizarposicion(mapa);
        posicionreal = Escena(posicion.X, posicion.Y, posicion.Z);
        posicionrealfutura = Escena(posicionfutura.X, posicionfutura.Y, posicionfutura.Z);
        distanciainicial = (posicionrealfutura - posicionreal).Length();
        desplaza = (posicionrealfutura - posicionreal) / speed;
        trans = true;
        starttime=tiempo;
    }
    else
    {
        currenttime = tiempo;
        if (actualizado == botonpulsado())&&((currenttime-starttime) >0.35))
        {

            actualizarposicion(mapa);
            posicionrealfutura = Escena(posicionfutura.X, posicionfutura.Y, posicionfutura.Z);
            distanciainicial = (posicionrealfutura - posicionreal).Length();
            starttime = currenttime;
        }

        distanciaactual = (posicionrealfutura - posicionreal).Length();
        if ((posicionrealfutura - posicionreal).Length() > 2)
        {
            if (distanciaactual / distanciainicial > 0.25)
                posicionreal += desplaza;
            else if (distanciaactual / distanciainicial > 0.125)
                posicionreal += desplaza * (float)0.75;
            else if (distanciaactual / distanciainicial > 0.075)
                posicionreal += desplaza * (float)0.5;
            | else if (distanciaactual / distanciainicial > 0)
                posicionreal = posicionreal + desplaza * (float)0.35;
        }
        else
        {
            posicion = posicionfutura;
            posicionreal = posicionrealfutura;
            trans = false;
        }
    }
}

```

-Es la parte de código que detecta el inicio del movimiento y pone en marcha las variables para dibujarlo

-Permite que si el jugador mantiene la tecla que inicio el movimiento avance una casilla cada 0,35 s

-Ralentiza el movimiento conforme se va completando el movimiento

-Una vez acabado el movimiento asienta al personaje en la posición exacta

Función auxiliares

Esta estructura impide que el usuario introduzca errores al pulsar varias teclas de movimiento.

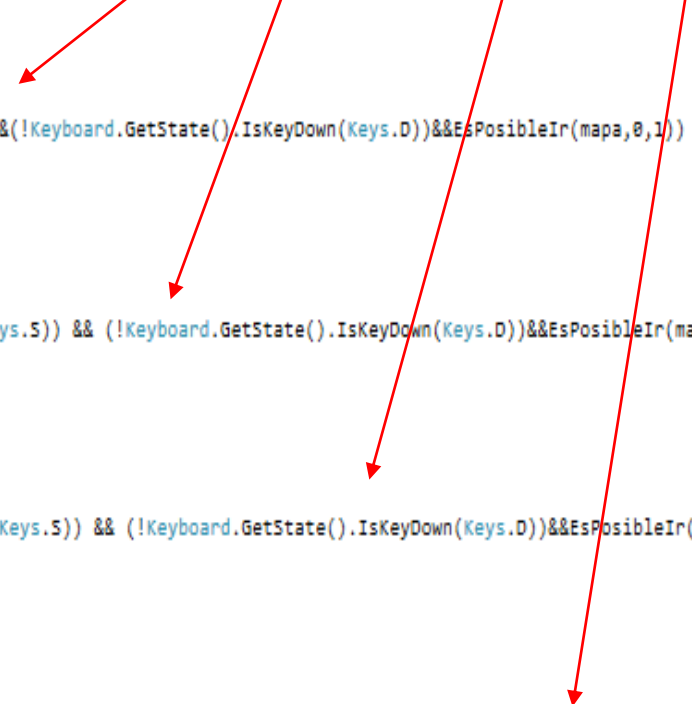
La principal de las auxiliares es : Actualizar Posición

```
void actualizarposicion(int[,] mapa)
{
    if (Keyboard.GetState().IsKeyDown(Keys.W) && (!Keyboard.GetState().IsKeyDown(Keys.A))&&(!Keyboard.GetState().IsKeyDown(Keys.S))&&(!Keyboard.GetState().IsKeyDown(Keys.D))&&EsPosibleIr(mapa,0,1))
    {
        posicionfutura.Y = posicionfutura.Y+1;
    }

    else if (!Keyboard.GetState().IsKeyDown(Keys.W) && (Keyboard.GetState().IsKeyDown(Keys.A)) && (!Keyboard.GetState().IsKeyDown(Keys.S)) && (!Keyboard.GetState().IsKeyDown(Keys.D))&&EsPosibleIr(mapa,-1,0))
    {
        posicionfutura.X = posicionfutura.X - 1;
    }

    else if ((!Keyboard.GetState().IsKeyDown(Keys.W)) && (!Keyboard.GetState().IsKeyDown(Keys.A)) && (Keyboard.GetState().IsKeyDown(Keys.S)) && (!Keyboard.GetState().IsKeyDown(Keys.D))&&EsPosibleIr(mapa,0,-1))
    {
        posicionfutura.Y = posicionfutura.Y - 1;
    }

    else if ((!Keyboard.GetState().IsKeyDown(Keys.W)) && (!Keyboard.GetState().IsKeyDown(Keys.A)) && (!Keyboard.GetState().IsKeyDown(Keys.S)) && (Keyboard.GetState().IsKeyDown(Keys.D))&& EsPosibleIr(mapa, 1, 0))
    {
        posicionfutura.X = posicionfutura.X + 1;
    }
}
```



Función auxiliar :botón pulsado.

Esta estructura impide que el usuario introduzca errores al pulsar varias teclas de movimiento.

```
int botonpulsado() //devuelve 0 si no hay tecla ,1 si es w ,2 si es A ,3 si es S, 4 si es D.
{
    int i =0;

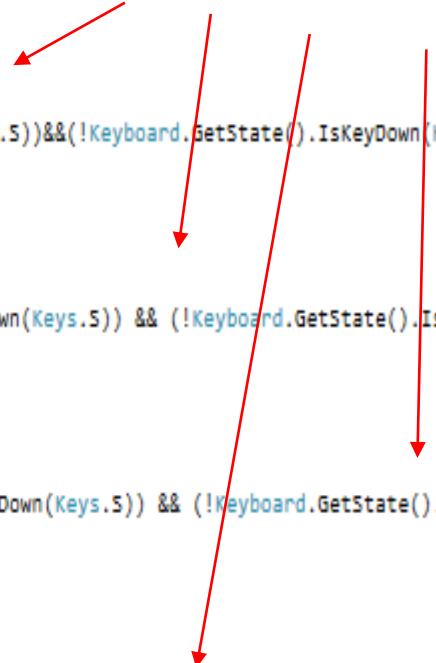
    if (Keyboard.GetState().IsKeyDown(Keys.W) && (!Keyboard.GetState().IsKeyDown(Keys.A))&&(!Keyboard.GetState().IsKeyDown(Keys.S))&&(!Keyboard.GetState().IsKeyDown(Keys.D)))
    {
        i = 1;
    }

    else if (!Keyboard.GetState().IsKeyDown(Keys.W) && (Keyboard.GetState().IsKeyDown(Keys.A)) && (!Keyboard.GetState().IsKeyDown(Keys.S)) && (!Keyboard.GetState().IsKeyDown(Keys.D)))
    {
        i = 2;
    }

    else if ((!Keyboard.GetState().IsKeyDown(Keys.W)) && (!Keyboard.GetState().IsKeyDown(Keys.A)) && (Keyboard.GetState().IsKeyDown(Keys.S)) && (!Keyboard.GetState().IsKeyDown(Keys.D)))
    {
        i = 3;
    }

    else if ((!Keyboard.GetState().IsKeyDown(Keys.W)) && (!Keyboard.GetState().IsKeyDown(Keys.A)) && (!Keyboard.GetState().IsKeyDown(Keys.S)) && (Keyboard.GetState().IsKeyDown(Keys.D)))
    {
        i = 4;
    }

    return i;
}
```



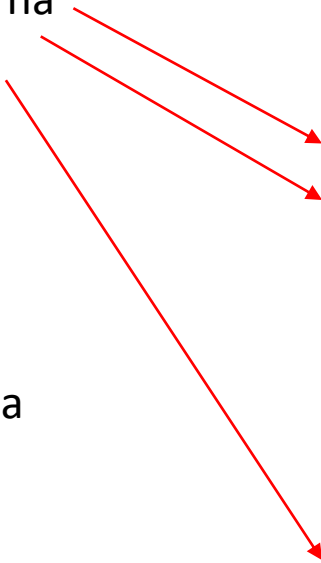
Choques y movimiento de cajas.

-La clase jugador tiene parámetros que almacena que a ocurrido un choque , donde ha ocurrido y en que dirección mover la caja.

-Estos parámetros son escritos cuando se intenta mover el personaje a través de la función actualizar posición.

-Son necesarias dos funciones para guardar la posición y la dirección del choque.

```
#region Parámetros
public Vector3 posicion;
public Vector3 posicionfutura;
public Vector3 posicionreal;
public Vector3 posicionrealfutura;
public Vector3 puntodechoque;
public Vector3 direcciondechoque;
Vector3 desplaza;
int ide;
float distanciaincial = 0;
float distanciaactual = 0;
float boxmesure;
public float speed;
bool trans=false;
public bool choque = false;
int actualizado;
float starttime;
float currenttime;
```



Funciones: choques y EsPosibleIr

```
bool EsPosibleIr(int[,] mapa,int x , int y)
{
    bool posible=false;

    if (mapa[(int)posicionfutura.X + x, (int)posicionfutura.Y + y] == 0 || mapa[(int)posicionfutura.X + x, (int)posicionfutura.Y + y] == 4)
        posible = true;
    else if (mapa[(int)posicionfutura.X + x, (int)posicionfutura.Y + y] ==2 )
        choques(posicionfutura, x, y);

    return posible;
}
```

```
#region choques
void choques(Vector3 punto,int x , int y)
{
    puntodechoque = punto;
    direcciondechoque.X = x;
    direcciondechoque.Y = y;
    choque = true;
}
#endregion
```

Una vez guardados estos parámetros en el método update se comprueba si se ha producido un choque y se procede al movimiento de la caja.

```
Jugador.MovimientoPersonaje(tiempo,map);  
if (Jugador.choque)  
    caja.Movimientocajas(ref Jugador.direcciondechoque, ref Jugador.puntodechoque, ref map, ref Jugador.choque);
```

En cada frame se ejecuta el movimiento de la caja de la misma forma que el del personaje .Cuando este acaba , la variable jugador. Choque cambia a false por lo tanto no actualiza más el movimiento.

Explicación del movimiento de la caja

```
public void Movimientocajas(ref Vector3 direccion, ref Vector3 punto, ref int[,] mapa, ref bool choque)
{
    if (trans == false)
    {
        actualizarcajas(direccion, punto, ref mapa);
        posicionreal = Escena(posicion.X, posicion.Y, posicion.Z);
        posicionrealfutura = Escena(posicionfutura.X, posicionfutura.Y, posicionfutura.Z);
        distanciainicial = (posicionrealfutura - posicionreal).Length();
        desplaza = (posicionrealfutura - posicionreal) / speed;
        distanciaactual = (posicionrealfutura - posicionreal).Length();
        trans = true;
    }
    else
    {
        if ((posicionrealfutura - posicionreal).Length() > 2)
        {
            if (distanciaactual / distanciainicial > 0.25)
                posicionreal += desplaza;
            else if (distanciaactual / distanciainicial > 0.125)
                posicionreal += desplaza * (float)0.75;
            else if (distanciaactual / distanciainicial > 0.075)
                posicionreal += desplaza * (float)0.5;
            else if (distanciaactual / distanciainicial > 0)
                posicionreal = posicionreal + desplaza * (float)0.35;
        }
        else
        {
            mapa[(int)posicion.X, (int)posicion.Y] = 0;
            mapa[(int)posicionfutura.X, (int)posicionfutura.Y] = 2;
            posicion = posicionfutura;
            posicionreal = posicionrealfutura;
            direccion = Vector3.Zero;
            punto = Vector3.Zero;
            choque = false;
            choquetrue = false;
            trans = false;
        }
    }
}
```

Las diferencias con el código de Personaje son :

- Solo se permite mover la caja una casilla
- La función es posible ir no tiene la función Choque.
- actualizar posición recibe como parámetros de entrada los valores del choque de personaje.

Funciones actualizar posición y EsPosibleIr de la clase Cajas

```
void actualizarcajas (Vector3 direccion ,Vector3 punto,ref int [,] mapa)
{
    if (EsPosibleIr(mapa, direccion, punto))
    {
        posicion = direccion + punto;
        posicionfutura = posicion + direccion;
        mapa[(int)posicionfutura.X, (int)posicionfutura.Y] = 3;
        mapa[(int)posicion.X, (int)posicion.Y] = 3;
    }

}

public bool EsPosibleIr(int[,] mapa, Vector3 x, Vector3 y)
{
    bool posible = false;

    if (mapa[(int)(2*x.X+ y.X), (int)(y.Y+2*x.Y)] == 0)
        posible = true;
    return posible;
}
```


Otras funciones:


Función ganar

```
void ganarjuego()
{
    if (map[12, 11] == 2 && map[12, 12] == 2 && map[12, 13] == 2 && map[12, 14] == 2)
    {
        ganar = true;
    }
    if (Keyboard.GetState().IsKeyDown(Keys.R))
    {
        Initialize();
    }
}

protected override void Draw(GameTime gameTime)
{
    GraphicsDevice.Clear(Color.CornflowerBlue);
    if (Jugador.choque)
        DibujarModelo(movilModel, caja.posicionreal);
    DibujarModelo(characterModel, Jugador.posicionreal);
    DibujarMapa();

    if (ganar)
    {
        spriteBatch.Begin();
        spriteBatch.DrawString(spFont, "Ganaste, presiona R para empezar de nuevo", new Vector2(200f, 250f), Color.Red);
        spriteBatch.End();
        GraphicsDevice.DepthStencilState = DepthStencilState.Default;
    }

    base.Draw(gameTime);
}
```



Esto de aquí impide
que se distorsione la
imagen

Vista general :

```
if (Keyboard.GetState().IsKeyDown(Keys.V))
{
    cameraPosition = Escena(8, 8, 0);
    cameraCentrado = Escena(8, 8, 0);
    cameraPosition.Z += 2000.0f;
}
else
{
    cameraPosition = Jugador.posicionreal;
    cameraCentrado = Jugador.posicionreal;
    cameraPosition.Z += 700.0f;
}
```

Funciones de Dibujo.

```
protected override void Draw(GameTime gameTime)
{
    GraphicsDevice.Clear(Color.CornflowerBlue);
    if (Jugador.choque)
        DibujarModelo(movilModel, caja.posicionreal);
    DibujarModelo(characterModel, Jugador.posicionreal);
    DibujarMapa();

    if (ganar)
    {
        spriteBatch.Begin();
        spriteBatch.DrawString(spriteFont, "Ganaste,presiona R para empezar de nuevo", new Vector2(200f, 250f), Color.Red);
        spriteBatch.End();
        GraphicsDevice.DepthStencilState = DepthStencilState.Default;
    }

    base.Draw(gameTime);
}

public void DibujarModelo(Model model, Vector3 position)
{
    Matrix[] transforms = new Matrix[model.Bones.Count];
    model.CopyAbsoluteBoneTransformsTo(transforms);
    foreach (ModelMesh mesh in model.Meshes)
    {
        foreach (BasicEffect effect in mesh.Effects)
        {
            effect.EnableDefaultLighting();

            effect.World = transforms[mesh.ParentBone.Index] * Matrix.CreateTranslation(position) * Matrix.CreateRotationY(boxRotation);
            effect.View = Matrix.CreateLookAt(cameraPosition, cameraCentrado, Vector3.Up);
            effect.Projection = Matrix.CreatePerspectiveFieldOfView(MathHelper.ToRadians(45.0f), aspectRatio, 1.0f, 10000.0f);
        }
        mesh.Draw();
    }
}
```

```

public void DibujarMapa ()
{
    for (int x = 0; x < mapWidth; x++)
    {
        for (int y = 0; y < mapHeight; y++)
        {
            if (map[x, y] == 1 )
            {
                DibujarModelo(boxModel, Escena(x, y, 0));

            }
            else

                if(map[x,y]==2)
                    DibujarModelo(movilModel,Escena(x,y,0));
                else
                    if((x==12)&&(y==14||y==13||y==12||y==11))
                        DibujarModelo(movilModel, Escena(x, y,-1));
                    else
                        DibujarModelo(sueloModel, Escena(x, y, -1));

        }
    }
}

```