2017

SIMULATION & PHYSICS – PRACTICAL 2

MOVEMENT

SJORS GIELEN
500765899

1. repeat the assignment you are implementing;

Create a ball which is set at the position of the mouse when the left mouse button is pressed

Create a spaceship which tracks a ball.
The spaceship requires the following behaviours:

- Acceleration towards the ball
- Easing when getting close to the ball

Create a "shield" object that follows the spaceship via elastic behaviour.

2. explain your approach;

The ball is extremely simple. It only requires a simple if input helper check along with ball.Position = inputhelper.mouseposition.

The spaceship has a number of approaches. I ended up doing two, first approach was a mostly "global" easing approach where the ship instantly rotated and had it's maximum velocity. However I changed everything to go towards a radians related approach. Now the ship's acceleration is always set towards the ball, the amount of acceleration is defined by the parameter of the spaceship instance.

The shield was extremely simple. I also added a "theather distance" too the shield so that the elastic behaviour only kicks in once the distance between it and it's target starts.

3. describe your code;

In the spaceship we first calculate the current distance between it and it's target via (this.position – target.position).Length(). This is followed by verifying if the current radians we are traveling at is the same as the goalRadians. If this is not the case we calculate our rotation amount for this frame, then the difference in angel between the goal and our current moving angel. If the absolute of difference in angel is the same or smaller then the amount we are allowed to rotate we simply set our radians to be the same as the goal radians. If this is not the case we verify with a simple if statement if we need to add rotation or reduces rotation. Afterwards we verify if the radians has remained within the -π to π range. If it is not we correct it.(Note that is is also done to the difference in angel)

Then we can simply calculate out the new acceleration value through use of cos and sin.
When the distance is smaller then 50 I decided to forcibly stop the ship. Using velicty.zero and acceleration.zero.
Otherwise when the ship is 200 distance away I start easing the ship. This is done by multiplying the velocity with the velocity.length / velocity.length * 1.05f
At last I update the angles of both the goal and the movementDir.
This would let me use my getRadians method on the movementDir to get the radians at witch the spaceship is moveing for the rotation of the ship.
For the shield object All I did was calculate out the distance of the itself and it's target. This is then compared with the theatherdistance. If the distance exceeds the theatherdistance we calculate the springing force via -k * (dist – theatherdistance).
Then the acceleration is simply in the direction of the theatherVector(this the normalized theatherVector) multiplied with the springing force and the gameTime.ElapsedGameTime.TotalSeconds (To un-bind it from the fps).

4. show (relevant) code snippets;

```
public float _Mass
{
    get { return this.mass; }
    set
    {
        if (value > 0)
        {
            this.mass = 0;
        } else
        {
            this.mass = value;
        }
        if (value < 0.5)
        {
            this.nimbelness = MathHelper.TwoPi;
        }
        else
        {
            this.nimbelness = ((value / (float)Math.Pow(value, 3)) * 90) * (float)Math.PI / 180;
        }
    }
}
```

spaceship mass property(to show how it binds in the nimbleness value)

```
public void updateAngle()
{
    //this.radians = Math.Atan2(this.Position.Y - target.Position.Y, this.Position.X - target.Position.X); //Point at stuff
    Vector2 tmp = Vector2.Normalize(velocity);
    if (!float.IsNaN(tmp.X))
        this.movementDir = tmp;
    tmp = Vector2.Normalize(target.Position - this.Position);
    if (tmp.X != float.NaN)
        this.goalDir = tmp;
    this.goalRadians = getRadians(this.goalDir);
    //this.radians = getRadians(movementDir);
}

public double getRadians(Vector2 vector, bool needvalue = false)
{
    if(vector.LengthSquared() == 0)
    {
        if (needvalue)
            return 0;
        return double.NaN;
    }
    return Math.Atan2(vector.Y, vector.X);
}
```

Update angle method and get radians method(found inside the spaceship class, but should really be moved to a more generic class)

```
ic override void Update(GameTime gameTime) {
Console.WriteLine(this.radians + "update" + u++);
//Step 2.6: Move the ship to the position of the target.
//position = target.Position;

float dist = (this.Position - target.Position).Length();

//Step 2.7: Comment step 2.6. Adapt the velocity to make the ship move to the target, include Easing.
if(this.radians != this.goalRadians)
{
    float rotation = this._Nimbelness * (float)gameTime.ElapsedGameTime.TotalSeconds;
    double differenceInAngel = this.goalRadians - this.radians;
    differenceInAngel = differenceInAngel < -Math.PI ? differenceInAngel + MathHelper.TwoPi : differenceInAngel > Math.PI ? differenceInAngel - MathHelper.TwoPi : differenceInAngel;
    if (Math.Abs(differenceInAngel) <= rotation)
    {
        this.radians = this.goalRadians;
    }
    else if(differenceInAngel < 0.0)
    {
        this.radians -= rotation;
    }
    else
    {
        this.radians += rotation;
    }
    this.radians = this.radians < -Math.PI ? this.radians + MathHelper.TwoPi : this.radians > Math.PI ? this.radians - MathHelper.TwoPi : this.radians;
    /*if (this.radians < -Math.PI)
    {
        this.radians += MathHelper.TwoPi;
    }
    if (this.radians > Math.PI)
    {
        this.radians -= MathHelper.TwoPi;
    }*///This comment is the same as the code shown on line 122
    Console.WriteLine(this.radians);
}
acceleration = new Vector2((float)Math.Cos(this.radians), (float)Math.Sin(this.radians)) * accel;
```

Spaceships update(1/2)

```
    acceleration = new Vector2((float)Math.Cos(this.radians), (float)Math.Sin(this.radians)) * accel;
    //Step 2.8: To speed up the ship, make it move three times as fast as before.

    //Step 2.9: Detect that the target position is reached and stop moving.
    if (dist < 50)
    {
        Velocity = Vector2.Zero;
        acceleration = Vector2.Zero;
    }
    else if (dist < 200 && !float.IsNaN(dist) && dist != 0 && velocity.LengthSquared() != 0)
    {
        velocity *= velocity.Length() / (velocity.Length() * 1.05f);
    }


    //Step 3.0: Compute the rotation of the ship.
    this.updateAngle();
    base.Update(gameTime);
}

//Step 3.1 Override the draw function to make the game draw the rotated spaceship.
public override void Draw(GameTime gameTime, SpriteBatch spriteBatch)
{
    spriteBatch.Draw(this.Sprite.Sprite, Position, null, Color.White, (float)getRadians(movementDir), this.Origin, .20f, SpriteEffects.None, 0f);
}
```

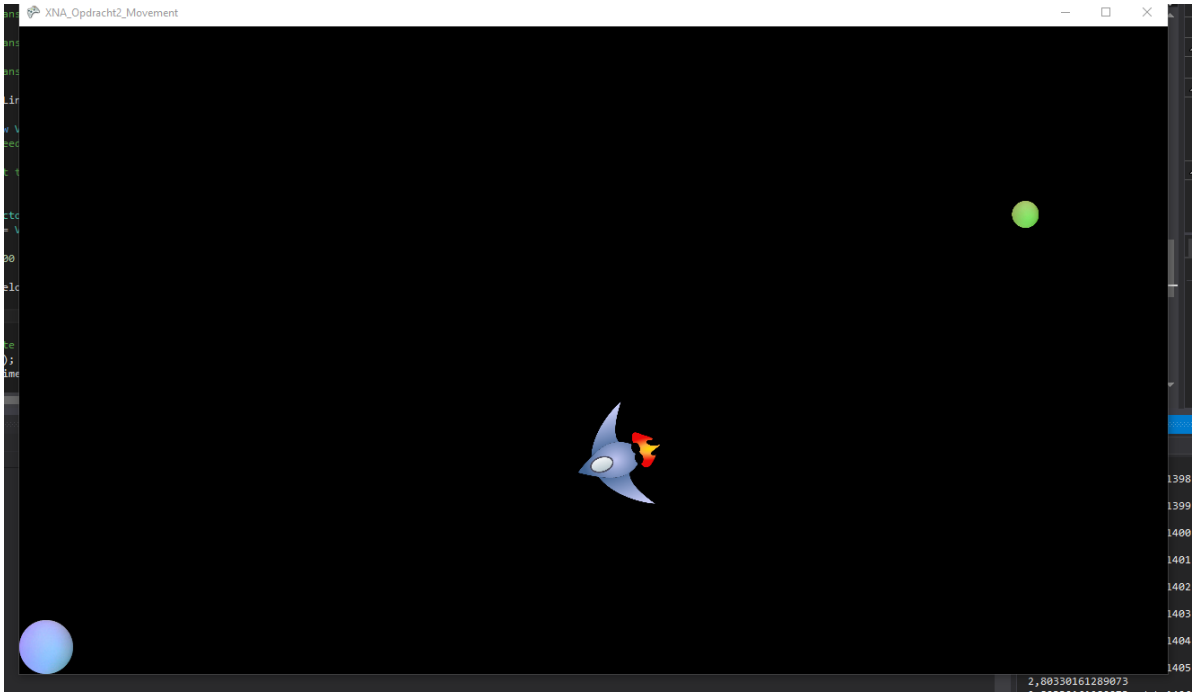
spaceship update(2/2)

```
public override void Update(GameTime gameTime) {
    //Step 4.3: Calculate springing force. F = -x * k => force = -displacement * k
    theatherVector = this.position - target.Position;
    float dist = theatherVector.Length();

    //Step 4.4 Use force as acceleration
    if (dist > theatherDistance)
    {
        float F = -springCoefficient * (dist - theatherDistance);
        acceleration = Vector2.Normalize(theatherVector) * F * (float)gameTime.ElapsedGameTime.TotalSeconds;
        //Console.WriteLine("shield accel :" + acceleration);
    }
    else
    {
        acceleration = Vector2.Zero;
    }
```

The shields update

5. include a screenshot of your program



Again I have made video's for the class as well These are found here: https://www.youtube.com/watch?v=De7I3tfr-FM&index=2&t=2040s&list=PLARkMALdMekM6EMkY0gcQSKvADVAx9zK5