Creating a Curved Trajectory

Matt Filer

For Angry Birds a curved trajectory is needed – which is based on the angle a bird is fired at, and the "energy" (in real terms) used by the player to pull it back in the slingshot.

Initially for this project I looked at a number of different approaches to this curve. First, I considered a quadratic curve.

$$-y = (\text{launch angle}) / x^2$$

This function gives a correct-looking trajectory and along with having the X axis it peaked at modified by the pullback energy given by the player it seemed convincing. Initial tests of this worked well but it did not produce the result I was aiming for, as the curve was not like the curve found within the real Angry Birds game, which I was interested in recreating.

I moved on to briefly look at Bezier curves however again quickly found that these did not produce the result I was after.

Eventually I landed on what is probably the simplest solution of them all – very basic gravity. To achieve this, I recorded a few parameters:

- Bird flight time (reset when the bird is loaded into the slingshot, counted from the moment the bird is fired to the moment it is destroyed or leaves the screen).
- Bird launch angle (based on the position of the bird when fired relative to the centre point of the slingshot).
- Bird launch power (based on the distance of the bird when fired to the centre point of the slingshot).

I then used this data to achieve the curve in two parts. By **subtracting** the pullback angle multiplied by a modifier from the Y value of the bird I create the effect of the bird being fired in the specified direction. Then, by **adding** the flight time multiplied by a modifier, the bird is slowly brought back down as if gravity is affecting it.

```
y = y + ((launch angle) * (modifier) * (frame time)) –
((flight time) * (modifier) * (frame time))
```

Of course this only gives us an up/down movement in the Y axis, so additionally the X position must be updated, which is simply done by using the launch power data multiplied by another modifier.

```
x = x + ((launch power) * (modifier) * (frame time))
```