Motion and Modelling - The Helicopter Screensaver

by Jelle Galgenbeld, Dmitri Kazachkov and Michel Lamoré

Motion and Modelling - The Helicopter Screensaver

The helicopter screensaver was made by adding several parts that we separately made and put those together. As a result this report is divided into three sections, one section for each member of our group. The division is as follows:

Jelle Galgenbeld – Code for the animation

Animation System

The animation system is a system used to make it look like objects are moving. For example the soldier who appears to be running. Of course you do not need to change the image every frame. So first we need to divide the usual time step into a time step that allows images to change at a lower rate.

```
void draw()
{
 if (step != int(frameRate)/10) {
  step += 1;
 }
 if (step == int(frameRate)/10) {
  step = 0;
 < ADD FUNCTIONS HERE>
 }
}
```

This is how we dealt with the time step problem. It works by having an integer count up to 1/10 of the framerate and then reset the counter. Now if you want to use this system you just place the function at the point where it resets, because that will only happen on each 10th frame.

That is the first part of the animation system. You still need to know what images you want to show when. To make this work we need to know how many images an animation has so it can start over when it reaches the final image to create a loop. To specify what animation is selected we need the image name (sprite_name) and we need to know what the object is doing at that moment (sprite_event). Then we need to specify the number of images required to create the entire animation. Then last, but not least we load the images.

```
 void display(String sprite_event_local, String sprite_name_local, int sprite_end_local)
 {
  sprite_event = sprite_event_local;
  sprite_name = sprite_name_local;
  sprite_end = sprite_end_local;

  if (sprite_event == "Right")
```

```
  {
    sprite_end = 1;
    HeliImg = loadImage(imagePrefix + sprite_name + sprite_event + sprite +".png");
  }
  if (sprite_event == "TurnLeftA" || sprite_event == "TurnLeftB" || sprite_event == "TurnLeftC" || sprite_event == "TurnRightA" ||
sprite_event == "TurnRightB" || sprite_event == "TurnRightC")
  {
    sprite_end = 1;
    HeliImg = loadImage(imagePrefix + sprite_name + sprite_event + sprite +".png");
  }

  if (sprite_event == "Left")
  {
    sprite_end = 1;
    HeliImg = loadImage(imagePrefix + sprite_name + sprite_event + sprite +".png");
  }
 }
```

This still does not give an animation, because it does not set the number of the image within the animation that needs to be shown. So we need a loop that cycles through the images in the order it needs to be played. So we need to count up to the last image of the animation and then start over again.

```
 void sprite(int sprite_local)
 {
  if (!sprite_activate)
  {
    sprite = sprite_local;
    sprite_activate = true;
  }
  if (sprite <= sprite_end) {
    sprite++;
  }
  if (sprite > sprite_end) {
    sprite = 0;
```

```
  }
 }
```

So now we have set the animation cycle and the only thing we need to do now is set the event for the object and it will display the right images to create an animation.

The Floodlight

The floodlight is made by creating a triangle from the helicopter to the place on the ground where the helicopter will be in 20 frames on the current speed, so when the helicopter turns around the floodlight will turn around with it without trouble. Also the floodlight sets the border for the helicopter to turn around in. In this case, where it has no target, the left and right border of the window.

```
    if (!target)
    {
     fill(0,255,0,50);
     triangle(x+dx*20, height, x, y, x+dx*20+100, height);
     leftborder = 160;
     rightborder = 1024 - 160;
    }
```

The floodlight described above only counts when there is no target for the helicopter. When it does have a target it will create a border set around the unit so that it will follow it until it is dead or in the anti-air gun.

```
if (target)
    {
     fill(255, 0, 0,50);
     triangle(floodlight_targetX-50, height, x, y, floodlight_targetX+20, height);
     leftborder = floodlight_targetX - 150;
     rightborder = floodlight_targetX + 150;
     bulAct=true;
     if (publicEnemy.x > width-50)
     {
      target=false;
     }
```

Those borders are also used when the helicopter has no target, they are at the edge of the screen then so when the helicopter knows he is approaching the border it will slow down and the turn animation is activated.

```
if (x > rightborder && dx > minSpeed) {
    dx -= .80;
  }
  if (x < leftborder && dx < maxSpeed) {
    dx += .80;
  }
```

This is the code that is used for slowing down the helicopter and then to change animation the helicopter will pick a new animation based on the speed it has.

```
if (x > rightborder && dx < maxSpeed*0.8 && dx > maxSpeed*0.4 && sprite_event != "TurnLeftA")
  {
    sprite = 0;
    sprite_event = "TurnLeftA";
  }


  if (x > rightborder && dx < maxSpeed*0.4 && dx > maxSpeed*-0.4 && sprite_event != "TurnLeftB")
  {
    sprite = 0;
    sprite_event = "TurnLeftB";
  }
  if (x >rightborder && dx < maxSpeed*-0.4 && dx > maxSpeed*-0.8 && sprite_event != "TurnLeftC")
  {
    sprite = 0;
    sprite_event = "TurnLeftC";
  }
```

This the code for when the helicopter is on the right edge of the screen and will turn left. The same thing is used for turning right, but then mirrored.

Dmitri Kazachkov – Pictures for the animation and the scenery

For the graphical part of our screensaver we have decided to use the sprite system, to draw the objects. Because no one of us had a real experience with any 3D modeling program, we used the models from a computer game (Battlefield 2). For this we used a program named ‘fraps’ to capture the screenshots from the game:

But of course we couldn’t just put those screenshots in our screensaver, so we had to edit them with Photoshop a little:

We took the screenshots from different viewing angels in the game, so later on we could use them in as an animation in our screensaver:

When the helicopter is reaching the edge of the screen, it starts changing the sprites in a way that it look like the helicopter is turning around.
Same principle we used for the animation of the explosion, we just got 6 pictures of different stages of a explosion which are changing each other in a certain order to make the explosion look realistic.

The background was made of three screenshots out of the game, 2 for the background itself and 1 for the bunker out which the soldiers are coming.

For the soldiers we used just 3 sprites that were changing each other in a certain time amount. 2 to gain the illusion of ‘a running soldier’ and the 3rd one to show a ‘dead soldier’ after his is being hit by

the helicopter. We planned to put an Anti-Aircraft tank in our screensaver, but due lack of time, we could only realize a gun of it in our screensaver.

In total we took more than 200 screenshots in the game and later have chosen the best of them to use in our screensaver. It could be easier if we had any experience with modeling programs, but as long as we didn't have this, we had to do it our way.

Michel Lamoré – Motion & shooting mechanism

Current realization

Soldiers

In our screensaver, the soldiers have to march from the left side of the screen to the right side to get to the anti-air gun, with which they can take down the helicopter if there are enough soldiers manning it. En route there is a chance they are killed by a bullet from the helicopter. To get to this animation we had to code several things. First of all, the soldiers are each given a different (random) speed, otherwise the helicopter would kill all soldiers with one successful attack, and on top of that, the viewer wouldn't be able to see the individual soldiers, as they would walk next to each other, which can't be seen in 2D. Instead the viewer would see just one soldier marching. Secondly, the soldiers walking on the ground can be shot at. To shoot the bullets at the soldiers, we first had the helicopter looking for a soldier with its floodlight, and when a soldier is seen by the floodlight, the helicopter starts tracking this soldier. Since we weren't able to create a proper “guided-bullet system”, where a bullet actively follows the path of the soldier to guarantee a hit, we fired 4 bullets in a random direction, 2 flying to the left and 2 flying to the right. The flightpath of these bullets is restricted, so that the highlighted soldier has the highest chance of getting hit, but it is however possible to kill a soldier who doesn't walk in the floodlight in our screensaver. Whether a soldier is hit by the bullet or not is decided by checking a range of 50 pixels (horizontally) around the soldier, at the height of his head. When the bullet is in this range, the soldier will die, and spawn again at the left of the screen after 3 seconds. Otherwise the helicopter keeps on firing until the highlighted soldier either dies or reaches the anti-air gun.

Anti-Air gun

When all soldiers have reaches the anti-air gun, this gun starts firing at the helicopter. A successful hit is once again determined by checking a horizontal range around the helicopter: in this case 200 pixels. This time however, a vertical range of 50 pixels is also checked, because the anti-air gun is immobile while the helicopter can move. When this vertical range would not be used, it would take a very long time before the helicopter is hit, making the screensaver boring. On bullet impact the helicopter gets a vertical speed, resulting in it “crashing” (actually flying to the ground). If the helicopter reaches the ground, it explodes, causing all soldiers on the anti-air gun to die, and after that the screensaver starts all over again.

Suggestions for improvement

Soldiers

Although each soldier has an individual speed, the movement of the soldiers can still be made more interesting. One possibility is to add in weather effects, such as snow, blocking the path of the soldier, or high temperatures, causing the soldier to slow down or even become dehydrated and faint.

Something we had in mind for this screensaver but couldn't add due to the limited amount of time we had for the project was different spawn possibilities. In the current situation the soldiers always come from the left of the screen, but it would be more interesting to have a jeep coming in that drops off soldiers, or soldiers coming from above with parachutes. Another thing which makes the screensaver more interesting is the ability of the soldiers to shoot at the helicopter while on foot. In this way, the helicopter will be in jeopardy all the time instead of only when the AA-gun is fully manned.

Bullets

Shooting bullets in a random direction is far from realistic, so when the development of this project is continued, a bullet that follows the path of the soldier would be a more realistic approach. Because this results in a very high hit rate, a certain random small error should be added to the direction of the bullet. In this way, it is not certain that a bullet actually kills a soldier. The same holds true for the bullet that is fired by the anti-air gun, which should also follow the path of the helicopter for a more realistic screensaver.

Outcome

Currently, the outcome of the on-screen battle is generally always the same: the soldiers march from left to right, some of them are killed, but because they always respawn, the anti-air gun will always be fully manned after a while, and the anti-air gun will always shoot down the helicopter in the end, causing the screensaver to restart. After a while this becomes boring to watch, as there is not a whole lot of variation. To make things more interesting, new helicopters and soldiers should randomly spawn (not only after other helicopters and soldiers have been killed), causing a non-predictable rise and fall in the number of helicopters and soldiers. In this way, you will never know how the battle will progress and what the outcome will be. To avoid an over-crowded screen the screensaver should reset itself after a certain period of time.

Interactivity

While adding interactivity turns the current project into something that can't be called a screensaver anymore, this project could be used as a starting point for a game. When the user can control the helicopter and shoot down soldiers, the screensaver can be turned into a 2D shooter, and the player can get points based on how many soldiers are hit. These points can then be used to for example destroy the anti-air gun, or to buy a new helicopter. We have actually converted the screensaver in its current form into a game, and the only thing we did was make the helicopter user-controllable with a joystick. This already proved to be great fun.