

Torque Game Builder – Whack-A-Mole Tutorial Part 3

Part 3 – Multiple Moles

Fixing our mole spawn

You might have noticed that sometimes two moles spawn in the same point. To avoid this the moleLevel::spawnMole() function has to be modified in the moleLevel.cs script file. It first looks for any respawnPoints free. If so, it continues and randomly picks a respawnPoint until it finds one that is not occupied. It spawns the mole as before and then marks the spawn point as occupied.

```
function moleLevel::spawnMole(%this)
{
    // if all spawn points are occupied don't spawn a new mole
    if( %this.spawnPointsOccupied == %this.respawnPointSet.getCount() )
        return;

    // find a respawn point
    %pointFound = false;

    // search until we find a respawn point
    while ( !%pointFound )
    {
        %respawnPointIndex = getRandom( %this.respawnPointSet.getCount()-1 );
        %respawnPoint = %this.respawnPointSet.getObject( %respawnPointIndex );

        if( %respawnPoint.isOccupied $= "" )
            %pointFound = true;
    }

    // select a color for the mole
    %color = getRandom(2);
    if( %color == 0 )
        %color = "Red";
    if( %color == 1 )
        %color = "Green";
    if( %color == 2 )
        %color = "Lilac";

    // create a new Mole
    %mole = new t2dAnimatedSprite()
    {
        sceneGraph = %this;
        class = "Mole";
        layer = 4;
        size = "16 16";
    };

    // play the right animation according to the color
    %mole.playAnimation("animMoleComeOut" @ %color);

    // save the selected color as dynamic field
    %mole.moleColor = %color;
}
```

Code continued on the next page

Torque Game Builder – Whack-A-Mole Tutorial Part 3

```
// set the position of the mole to the position of the respawn point
%mole.setPosition( %respawnPoint.getPosition() );

// mark this respawn point as occupied
%respawnPoint.isOccupied = true;

// increment the occupied counter
%this.spawnPointsOccupied += 1;

// give the mole a reference to its respawn point
%mole.respawnPoint = %respawnPoint;
}
```

Code Sample 3.1

We also have to mark the respawn points as free again when a mole gets whacked (in the mole.cs),

```
function mole::onAnimationEnd(%this)
{
  if(%this.getAnimationName() != ("animMoleComeOut" @ %this.moleColor))
  {
    // free the respawn point for other moles
    %this.respawnPoint.isOccupied = "";

    // decrement the occupied counter
    %this.sceneGraph.spawnPointsOccupied -= 1;

    %this.sceneGraph.schedule( 1500, "spawnMole");
    %this.safeDelete();
  }
}
```

Code Sample 3.2

And we have to set the occupied counter to zero when the level is loaded. Therefore we modify moleLevel::onLevelLoaded() (in moleLevel.cs):

```
function moleLevel::onLevelLoaded(%this)
{
  %this.respawnPointSet = new SimSet();

  // set the occupied counter to zero
  %this.spawnPointsOccupied = 0;
}
```

Code Sample 3.3

And we better remove the moles that we created in the level builder. Because they were not created through moleLevel::spawnMole() it will mess up the occupied counter.

Torque Game Builder – Whack-A-Mole Tutorial Part 3

Automatic Moles

Now there are no more moles in level builder. They should be created automatically. To accomplish this we again write a function that does this for us. Put this in moleLevel.cs at the beginning of the file:

```
// values in milliseconds
$MOLE_FACTORY::maxCreationInterval = 2000;
$MOLE_FACTORY::minCreationInterval = 400;

$MOLE_FACTORY::speedUp = 30;

function moleLevel::startFactory(%this)
{
    // spawn a new mole
    %this.spawnMole();

    // calculate when to spawn the next mole
    %delay = %this.minCreationInterval + getRandom(%this.maxCreationInterval - %this.minCreationInterval);

    // call the function recursively but with the calculated delay
    // we save the event id so we can stop the function if we want
    %this.factoryEventId = %this.schedule( %delay, "startFactory" );

    // calculate new creation intervals so the game gets faster
    %this.minCreationInterval -= %this.speedUp;

    if(%this.minCreationInterval < 1)
        %this.minCreationInterval = 1;

    %this.maxCreationInterval -= %this.speedUp;

    if(%this.maxCreationInterval < 1)
        %this.maxCreationInterval = 1;
}

function moleLevel::stopFactory(%this)
{
    cancel( %this.factoryEventId );
}
```

Code Sample 3.4

By modifying the values at the top the game can be made faster or slower. Now the factory has to be started when level is loaded and stopped when the level is unloaded. So we modify moleLevel::onLevelLoaded and onLevelEnded again:

Torque Game Builder – Whack-A-Mole Tutorial Part 3

```
function moleLevel::onLevelLoaded(%this)
{
    %this.respawnPointSet = new SimSet();

    // set the occupied counter to zero
    %this.spawnPointsOccupied = 0;

    // load the start values
    %this.maxCreationInterval = $MOLE_FACTORY::maxCreationInterval;
    %this.minCreationInterval = $MOLE_FACTORY::minCreationInterval;
    %this.speedUp = $MOLE_FACTORY::speedUp;

    // start the mole creation
    %this.startFactory();
}

function MoleLevel::onLevelEnded(%this)
{
    // stop the mole creation
    %this.stopFactory();

    %this.respawnPointSet.delete();
}
```

Code Sample 3.5

Try it!

Mole Extension

Until now a mole could only disappear if it was whacked. However, we want them to dive back into the ground if they have got enough fresh air. Therefore we schedule them to dive back in when they are created. We need to create a new functions in mole.cs and modify an existing one:

```
//This function will let the mole dive into the earth again but only if it was not already whacked.
function mole::diveIn(%this)
{
    // only if not already whacked
    if( %this.getAnimationName() $= ("animMoleComeOut" @ %this.moleColor) )
        %this.playAnimation("animMoleDiveIn" @ %this.moleColor);
}

function Mole::onAdd(%this)
{
    %this.setUseMouseEvents( true );
    // let them dive back into the earth after 1 second
    %this.diveInEventId = %this.schedule(1000,"diveIn");
}
```

Code Sample 3.6

We schedule the retreat in the Mole::onAdd() method that is called on when the mole is created. We save the event ID so we can cancel the event when the mole gets deleted. Otherwise there will be problems if an event is called on an object that doesn't exist anymore.