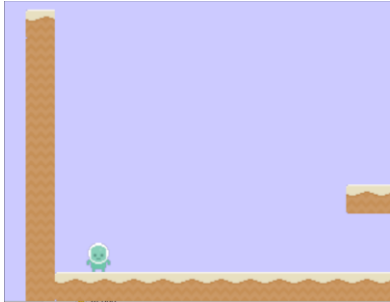


## Game Programming: Exercise 10: Animations

|                     |  |
|---------------------|--|
| Learning objectives | <p>Learning objectives</p> <ul style="list-style-type: none"> <li>• Implement Cel animations / sprite animations based on physics simulation</li> <li>• Implementing easing animation as an improvement to simple linear interpolation.</li> <li>• Implement spline curves for advanced movement</li> <li>• Implement a simple animation script using Lua</li> </ul> <p><b>Handing in:</b> Individual hand in. Create a <b>zip</b>-version of source-files, header-files and resources (CMakeFileLists.txt, .json and png). We will build your project using CMake, so make sure it works before hand-in (Note: If you make sure to keep all files in the same directory it should work without you need to change the CMakeFileLists.txt).</p> <p><b>Controls:</b><br/> Left / Right – controls the character<br/> Space - Jump<br/> D – debug menu and visualize fly movement path<br/> Z – zoom</p>  |
| 10-1                | <p><b>Cel animations</b></p> <p>Implement cel animations of the main character, by implementing the function <code>CharacterController::updateSprite()</code></p> <ul style="list-style-type: none"> <li>• If the character has zero velocity, then the standing sprite must be used.</li> <li>• If the character is grounded while moving, then the walk1 and walk2 sprites must be used. Choose an appropriate animation speed based on the horizontal velocity. Animation speed should change proportionally to horizontal velocity.</li> <li>• If the character jumps, use the sprite flyUp, fly and flyDown for the three phases of the jump.</li> <li>• The sprite must be flipped around the x-axis if the character moves towards left.</li> </ul>   |
| 10-2                | <p><b>Platform moving using easing functions</b></p> <ul style="list-style-type: none"> <li>• Choose a non-linear easing function to move the platforms in <code>MovingPlatformComponent::update()</code></li> </ul>   |

|             |  |
|-------------|--|
| <b>10-3</b> | <p><b>Bird Movement using Quadratic Bézier curves</b></p> <ul style="list-style-type: none"> <li>• The bird (or fly?) moves along a sequence of predefined points. Intermediate positions are defined by linearly interpolating between pair of points using <code>glm::mix()</code> in the <code>BirdMovementComponent::computePositionAtTime()</code> method.</li> <li>• Replace the linear movement trajectory with Quadratic Bézier curves. Note that you also need to change <code>BirdMovementComponent::getNumberOfSegments()</code> to return the correct number of segments.</li> </ul> <p>The game after step 10-3 can be seen here:<br/> <a href="http://www.itu.dk/~mnob/platformer/platformer.html">http://www.itu.dk/~mnob/platformer/platformer.html</a></p>  |
| <b>10-4</b> | <p><b>Spiral Bird Movement using Script Component</b></p> <ul style="list-style-type: none"> <li>• Uncomment the block comment in <code>PlatformerGame::initLevel()</code> line 129-159</li> <li>• In <code>ScriptComponent::init()</code> extend <code>GameObject lua</code> definition to include the functions needed to run the lua script from <code>PlatformerGame::initLevel()</code>. <ul style="list-style-type: none"> <li>◦ Hint: look at the errors in the debug console, when the lua is executed.</li> </ul> </li> <li>• Once you have the second bird flying in circles, change the lua script in <code>PlatformerGame::initLevel()</code>, to update radius continuously so that the bird flies in a spiral with radius between [10, 100]</li> <li>• Example of radial movement over time (i.e. 100-&gt;10-&gt;100)</li> </ul> <div data-bbox="624 1034 1295 1350"> </div> |