Documentation

Group/Game Name: Tomb saviour

Brief description of implementation:
Story: Tomb raiders are in the process of robbing the tomb of Ramesses II. The first thing they did was to take the sarcophagus out of the pyramid and now they are looking for more treasures. Meanwhile, the mummy of Ramesses II. comes back to life through Osiris, the god of the dead, to take revenge on the tomb robbers. His greed for vengeance is only fulfilled when the last thief has been eliminated and he has returned to his pyramid, and so the mummy makes its way into the pyramid.
The video game was created with the libraries below in C++ and OpenGL. Details of the effects used and the design of the gameplay can be found beneath.

Additional libraries:
Physx – to simulate gravity, include basic and advanced physics like collision detection for static and dynamic objects (e.g. static cacti, dynamic enemy)
https://github.com/NVIDIAGameWorks/PhysX
Assimp – to import 3D objects (.obj files):
https://github.com/assimp/assimp/releases
OpenCV – to load a video:
https://opencv.org/

Gameplay:

Mandatory:

- 3D Geometry: We create objects from .obj files, and animated Models from .dae files, for our game. Objects.cpp is our general class for generating the objects (both mesh model and then the physx model in the physx scene). In model.generateModel(path) the path to the object is passed along. In this method it is also visible that we use the assimp library to read the files. In our physx scene static objects are created from their vertices and indices and dynamic objects from their size (Bounding box).
- Playable: Like in a real game you can walk on the ground (see also Collision Detection), you can collect objects and shoot them. The game can also be started via the exe. To ensure a real game experience, the player has lives (scarabs), which one will lose if you are touched by the tomb raider(s). In addition, the shot spikes have an effect on the enemies so that you can defeat them. (See also Win/Lose Condition)
- Advanced Gameplay: Since our game is kind of a shooter game, there is not much to add to the playable section.
- Min 60 FPS and Framerate Independence: We included the independent framerate in the render loop. In Main.cpp in the render loop objects.render is called and in this method at the end: physxscene->simulate(...). The deltatime is also passed to the controller or dynamic objects (enemy&character) and multiplied by the movespeed.

- Win/Lose Condition: The player has lives which are lost if touched by one of the tomb raiders. At the last live deduction the game is over. If you manage to get to the treasure chest in the pyramid with at least one life left and all tomb raiders dead, you win.
- Intuitive controls: The player controls are explained below, in general we use the same controls as in other games. In Character.cpp the polling events are built in (coming from object.render(...)) for the movement with WASD since pressing those keys should have a continuous effect (also the camera movement is solved with polling). On the other hand, we use callbacks for single events like wireframe toggling or (de)activate Fullscreen mode.
- Intuitive Camera: The camera can be moved intuitively by moving the mouse. There are different approaches, but we have chosen two. On the one hand, when the mouse is at the edge of the window, the view slowly rotates. On the other hand, with the second camera control, you can move the mouse beyond the window, and it automatically lands back in the center of the window. (Change by pressing C, shooter camera needs to be modified a bit)
- Illumination model: We have implemented two lights (point&directional) for every object (see Shader.cpp, create PhongShader). We are using a phong illumination model for the objects outside of the Pyramid. See effects for the Simple Normal Mapping.
- Textures: In Texture.cpp the texture is read from a path to a dds file and then assigned to the corresponding object. Every object has a texture. For the enemy the Texture is created inside the Model.cpp class because the enemy.dll contains texture information.
- Moving Objects: The tomb raiders as well as the shot spikes are the moving objects in our game. In enemy.move(...) the position of the player is taken to determine the direction of movement for the enemy. The flight direction for the thrown spikes is calculated in PhysxScene.throwSpikes(..) by taking the camera view direction.
- Documentation: See this document.
- Adjustable Parameters: The window is started with 800*800 dimensions, but it can be made wider/narrower/higher/lower at any time. Fullscreen-mode is also possible (see user interaction), in Callbacks.cpp there is a method that listens for F5 and when pressed sets the window to fullscreen by taking the dimensions of the current screen and adapting the window with it. The refresh_rate is set in Main.cpp, but can be adjusted via the settings.ini file. At the beginning the illumination multiplier gets assigned to the shaders/lights.

Optional:

- Collision Detection (Basic Physics): As we use the physics engine PhysX, each object has its own bounding box for collision. The moving character is a PhysX controller with the camera fixed on top. Accordingly, Character.cpp and Camera.cpp work together where the camera determines the view direction and thus the walking direction and in Character move(...) the controller (and thus the camera) is moved by a collide-and-slide algorithm.

- Advanced Physics: Our opponents are also physx controllers, which makes them dynamic and gives them their own bounding box. This allows us to determine when the opponent collides with the mummy through a PxControllersHit Callback. The Collision between spikes and the mummy is detected through a PxSimulationEventCallback in physxScene.cpp. In Objects.cpp the opponent gets the current position of the mummy in draw(...) to be able to follow the player. Other dynamic objects are the spikes that can be shot. (There will maybe be changes here until the 2nd delivery).
- Heads-up Display: The HDU displays how many lives the player has left (bottom left), how many shots he can still fire (top right) and at the beginning and end of the game an instruction menu and an end message (win/lose) is shown. This is created by a very thin object and the appropriate texture (see Hdu.cpp).

Effects:

Lighting:

- (Shadow Map with PCF): We implemented the basics for the shadow map and had some good results but in the end we couldn't create a shadow for every object and skipped it.

Terrain:

- Tessellation from Height Map: In Objects there is a createTerrain(...) method. It first creates a terrain shader and then creates a terrain mesh model from a heightmap png file. After that the same terrain is created again as physx object/actor by creating an actor with a heightfield geometry so that the player can not only see the terrain but also walk on it with the controller.

Animation:

- GPU Vertex Skinning: The vertex skinning is used for the enemies. For this we pass in Object::createEnemy() in the generateModel(..) method the path to a .dae file, so that also the bone infos can be read out. The .dae also contains informations about the texture (path, coordinates) which are being read out and processed in the Model::loadTexture() function. By loading the animations of Punching.dae and the Dying.dae the default animation from the model is changed when the enemy is close enough to the mummy or was hit by a spike.

Texturing:

- Video Texture: We created a Videowall which displays the a video on the ceiling inside the pyramid. [most important methods for this: Objects::createVideoWall(), Shader::loadVideoTexture(), Shader::getcurrentFrameIndex()], can be turned on/off through 0/1 in settings.ini

Shading:

- Simple Normal Mapping: This effect is used as an effect on the treasure chest inside the pyramid. One can turn on/off the effect with “N”.

Other special features:

- Different enemy animations
- Change mummy’s life in settings.ini
- Two Camera modes
- Background music + sound effects
- Difficulty level (how fast the enemy deduct one life)

Walk-through:
As the player, you control the mummy whereby the goal is to get inside the pyramid and eliminate all tomb raiders. You can collect cacti and fire the spikes at your enemies. You win if you make it into the pyramid (touch the treasure chest) and all the tomb raiders have been killed.

User interaction	• Mouse: Camera view direction • WASD: Movement of Mummy (depending on the view direction) • Space: toggle super speed for mummy movement (maybe space is later used for jumping, this is not yet decided) • 0: shoot spikes • 9: collect cactus • C: change camera control (take the control you like best!) • B: *Cheat* get back to the start position • N: toggle Simple Normal Mapping effect • LMB (left mouse button)/RMB: Collect objects / Shoot spikes • ESC: Quit the game • F1: toggle wireframe mode • F2: toggle backface culling mode • F5: toggle fullscreen mode