

Documentation

Group / Game Name:

Defender Chick

Brief Description:

We implemented Defender Chick based on the GCG vulkan framework this course provided. In the game you must collect all the chickens and bring them to the stable. By default, one chicken is animated using vertex skinning. All other chickens (and also the squirrels) have contours via backfaces. At the same time, Squirrels will try to get to the stable (moving objects). If the squirrels are not hindered from reaching the building, e.g. by carrying them away, the game is lost. If all the chickens are inside the stable, and no squirrel has reached it, the game is won. This mechanic uses PhysX trigger zones, as further described in the advanced physics gameplay description. There is also a timer implemented in which time all chickens have to be collected to win the game, visible as output in the terminal.

We implemented all the mandatory gameplay, as well as basic and advanced physics and sophisticated gameplay. From the effects list, we implemented vertex skinning (animated chicken), a shadow map with PCF (for the whole static scene, e.g. the barn and trees) and contours via backfaces (for non animated chickens and squirrels). We also implemented our own geometry loader using the tinyglTF library. All effects and gameplay implementations should be easily found. It might take some time to find the animated chicken, but the amount is changeable in the window.ini. Though performance is reduced when spawning more animated chickens, which might lead to lower FPS count.

Using Vulkan was the right choice for us, even though it sometimes was really challenging. We are really proud of the game we created.

Additional libraries:

- tinyglTF (<https://github.com/syoyo/tinygltf>)
used in our GeometryLoader to load glb models
- PhysX (<https://github.com/NVIDIA-Omniverse/PhysX>)
used in our PhysicsEngine for basic physics, trigger zone, gravity, picking up objects, throwing objects, collision detection, ... - for building the program, the necessary .dlls for debug and release can be found in assets/dll/. These should be copied into the corresponding build/* folder, where the executable is located.

Geometry:

- <https://www.cgtrader.com/free-3d-models/exterior/landscape/village-southern-europe-low-poly>
(environment)
- <https://sketchfab.com/3d-models/squirrel-f442c20ca7dd4a988a17a12604556672> (squirrel)
- <https://sketchfab.com/3d-models/chicken-walkcycle-17a2a56e1ea04288895d17075442b305>
(chicken)

Mandatory:

- **3D Geometry:** chicken, squirrels, building, trees, stones - all objects were loaded from an external file (glb, links above under "Geometry")
- **Playable:** collect the chickens and carry them to the structure in the center of the map, keep squirrels away from the structure, submission startable via .exe file
- **Min 60 FPS and Framerate Independence:** In tutor hours FPS were above 60, all movements are framerate independent (divided by delta time)
- **Win/Lose Condition:** WIN: Collect and bring all chickens to the central structure in the given time. LOSE: Squirrel gets to the central structure or time runs out. The time left, counter of the chickens inside the barn as well as the result (win/loss) of the game is output in terminal.
- **Intuitive controls:** ESC - close game, F1 - polymodes, F2 - cullmodes, W - move forward, E - pick up object (ray cast look at direction), R - restart the game, A - move left, S - move back, D - move right, LEFT SHIFT - sprint, SPACE - jump, MOUSE - camera orientation
- **Intuitive Camera:** 1st person camera attached to PhysX player controller. The controller can move freely around the map.
- **Illumination model:** Directional light coming from above, all objects use the phong shading from the framework and have a material assigned to them. Also, each object has normal vectors.
- **Textures:** Squirrels use texture with mipmapping and trilinear filtering.
- **Moving Objects:** Squirrel moves on its own, towards the barn. Chickens and squirrels can also be picked up and moved around.
- **Documentation:** this
- **Adjustable Parameters:** Parameters can be adjusted using the window.ini file in bin/assets/settings/window.ini (screen resolution (window_width, window_height), fullscreen, number of objects, time used for timer)

Optional:


- **Advanced Gameplay:** Squirrels spawn randomly and always move towards the barn, if it reaches the barn, it will restart the game. This adds another loss condition and makes the game much more interactive and fun, since it adds a secondary mission to the game, keeping the squirrels away from the barn.
- **Collision Detection (Basic Physics):** Collision detection between character controller and static environment, so the player can not move through trees or stones, as well as jumping and sliding off slopes (stones).
- **Advanced Physics:** We implemented a trigger zone for counting the amount of chickens in the structure. The trigger zone also reacts to squirrels entering it by restarting the game. It is also possible to pick up (using the E button) and throw chickens and squirrels.

Effects:

Lighting:

- **Shadow Map with PCF:** shadow map with PCF for the whole static scene (e.g. environment) - shadows are also cast on moveable objects (which are not included in the shadow map, so they themselves can not cast shadows) - PCF is applied in the fragment shaders. We create the shadow depth image of the scene once before the while loop, using an offscreen renderpass. In each iteration, the image is then submitted to the queue.


Resources: We took inspiration from the PCF functions from the github for our shader. We built our own shadow offscreen renderpass inspired by the implementation on github.

- <https://github.com/SaschaWillems/Vulkan/blob/master/examples/shadowmapping/shadowmapping.cpp>
- <https://github.com/SaschaWillems/Vulkan/tree/master/shaders/glsl/shadowmapping>
-  C/Vulkan Game Engine Dev Ep. 153 Finishing Shadow Mapping (for now) !tts !kofi !m...

Animation:

- **GPU Vertex Skinning:** vertex skinning is applied to one of the chicken. The model's bone structure is loaded from the .glb file and applied to it, based on the bone's weights. The skinning itself is calculated on the GPU side (texture_skinned shaders). The animation is looped and updated per frame in the render loop.

Resources: We took a lot of inspiration from the github, reduced it to what we need, adapted the GCG framework functions.

- <https://github.com/beaumanvienna/vulkan/tree/master/engine/renderer/skeletalAnimation>
-  Skeletal Animation with Vulkan, C++, and glTF plus setup with Blender and Mixamo

Post Processing:

- **Contours via Backfaces:** Implemented contours via backfaces for non animated moveable objects (non animated chickens and squirrels). Wireframe mode for contours via backfaces is disabled, so when pressing F1 in front of a non animated object the scaled up, darker backface is visible.

Resources: we just gathered an overview, implemented the contours by creating an adapted version of the pipeline we already had in the framework.

- <https://github.com/SaschaWillems/Vulkan/blob/master/examples/occlusionquery/occlusionquery.cpp>

Defender Chick

Graphics Card: NVIDIA GeForce GTX 1060 6GB



Stefan Samer | e12120498 | UE 033 532

Lucia Kloos | e12122563 | UE 033 532

Other special features:

- **Model Loader:** we implemented our own model loader which uses tinyglTF to load external assets such as geometry, textures, normals, and skeletal animations. (GeometryLoader.h / .cpp)
- **Respawn Logic:** should for some reason a chicken, the player, fall off the platform or get outside the scene, it will respawn. The player will be back at its starting position, chickens will respawn randomly.

Resources: inspired the model loader by the github implementation and adapted it to fit the geometry class from the framework.

- <https://github.com/syoyo/tinyglTF/tree/release>
-  Vulkan Tutorial #021 [Deutsch] Laden und Rendern eines GLTF Meshes
-  [mini 02] glTF and FBX loading (C++)