

Administrative	
Group Name/ Game Name	Super Nicer Dicer
GitHub Link	https://github.com/michael-rubik/cgue21-SuperNicerDicer
Students	Martin Gerdenich, 11776233 Michael Rubik, 11809937
Genre	Jump & Run
Goal	Collecting all 5 Playing Cards before time runs out
Gameplay	
3D - Geometry	We import our objects from Blender ¹ with Assimp-Modelloader. Using the Assimp Library ² we get all the object information we need, like Normals/UV etc.
Playable/Advanced Gameplay	Our player can be controlled and when colliding with cards the player collects the card. Furthermore the cube can collide with the obstacles like trees/cliffs etc.
Min. 60 FPS and Frame - Independence	We use the deltaTime for all relevant calculations like the character moving to insure Frame-Independancy.
Win/Lose Condition	Our game is working with a one minute timer. The player has to collect all the cards in the level within the time limit or the game is over.
Intuitive Controls	The player is controlled via the WASD keys. Controls are implemented via polling events. The player also able to jump with the Space Bar.
Intuitive Camera	The camera is behind our player and with the mouse you are able to rotate the camera to the left or right.
Illumination Model	We have a directional light as lightsource which moves above the playing field and the objects have materials and normals.
Textures	Our models which are all imported with Assimp have a texture which are imported aswell. Furthermore we use a normal map which is also imported with the model and a lightmap which is loaded in and passed to the modelShader to mix the diffuse Texture of the model with the lightmap to get the shadows from the lightmap.
Moving Objects	We have the player-character who can move and a moving light to make sure the surface slightly changes because of simple normal mapping.

¹<https://www.blender.org/>

²<https://www.assimp.org/>

Adjustable Parameters	The parameters can be changed in the settings.ini file so the program does not have to be recompiled.
Optional Gameplay	
Collision Detection and Advanced Physics	We have used the bullet engine to make collision possible and move the player. Our player has a rigid body and can collide with the other objects like the Cliffs/Trees and also the cards. In case of a collision with a card it triggers the collection event.
Effects	
Lightmapping using separate Textures	For lightmaps ³ I firstly built the entire Level in Blender and added a Sunlight and a direction so all my objects throw a shadow. Then baking the light to a separate texture which is my lightmap. Renderbaking ⁴ is used to make Textures like colors, normal maps and lightmaps which can then be used in the game. In our game i load the lightmap and pass it to the modelshader. In the modelshader the lightmap is then mixed with the diffuse texture. You can mix them in the shader with this command: <pre>„mix(texture(texture1, TexCoords), texture(texture2, TexCoords), 0.2)“.</pre> The first two arguments are the textures with the corresponding Texture Coordinates and the last argument is a factor which is the mixing ratio.
Simple Normal Mapping	For the simple normal mapping ⁵ we made our dice in blender and connected the diffuse and normalmap. When exporting the object a MTL File is created which is used to load the model with the diffuse and normal map. The calculations to make Simple Normal Mapping work on our player character is done in the normalmapping shader.
CPU Particle System	For the particle-system ⁶ we used instancing. Each particle is a simple mesh which is a quad made of 2 triangles, and we have several instances of these. We have buffers for the vertices of our Mesh, the center of those particles and color. And to draw our particles we use the function glDrawArraysInstanced.
Bloom/Glow	Bloom ⁷ gives the bright regions of our scene a glow effect. In our case we used this effect to

³<http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-15-lightmaps/>

⁴<https://docs.blender.org/manual/en/latest/render/cycles/baking.html>

⁵<https://learnopengl.com/Advanced-Lighting/Normal-Mapping>

⁶ <https://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/>

⁷<https://learnopengl.com/Advanced-Lighting/Bloom>

	make the cards glow and bloom.
Video Texture	Extracted the frames from a short video, display and update them periodically to show a continuous video. Basically a big screen on the Edge of the map.
Extra Inputs	
WASD	Control the player (in Debug Mode the camera)
Mouse moving	Moves the camera
Spacebar	The player jumps
F3	Toggle Debug Camera
F9	Toggle Normal Mapping
ESC	Quit Game
Additional Information	
Used Libraries	Bullet, Assimp, glew, glfw, glm (ECG library), stb_image
How To Play The Game	
<p>After starting the game by executing the .exe you have to run around the map and collect all the cards within the 1 Minute time limit. Otherwise the game is over and in the Commandline you will see that you lost the game. If you manage to collect all the cards within the limit the Commandline will tell you you won the game. Have Fun!</p>	
Sources	
<ol style="list-style-type: none"> 1. https://www.blender.org/ 2. https://www.assimp.org/ 3. http://www.opengl-tutorial.org/intermediate-tutorials/tutorial-15-lightmaps/ 4. https://docs.blender.org/manual/en/latest/render/cycles/baking.html 5. https://learnopengl.com/Advanced-Lighting/Normal-Mapping 6. https://www.opengl-tutorial.org/intermediate-tutorials/billboards-particles/particles-instancing/ 7. https://learnopengl.com/Advanced-Lighting/Bloom 	