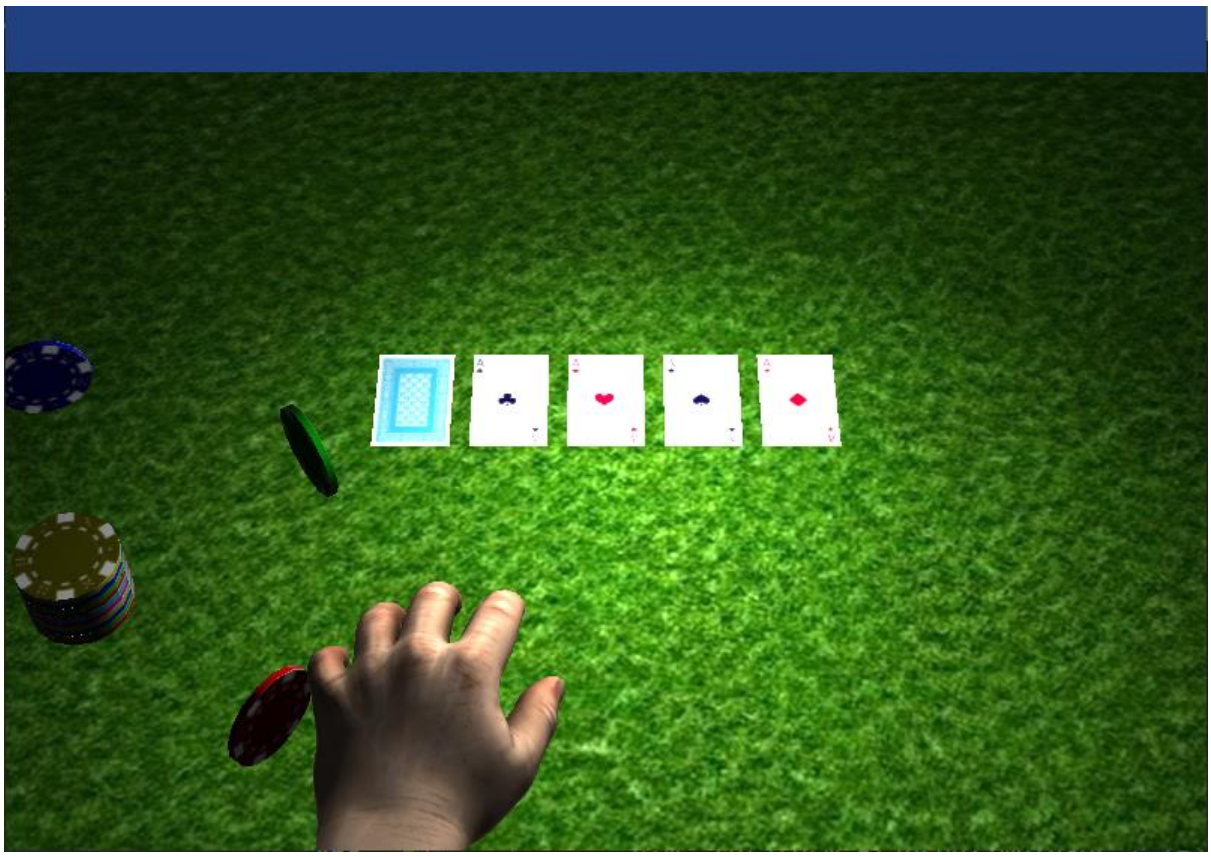
Computer graphics 2024 fall – Casino

Maximum achievable score: **112 points**.



0. Basics **(total of 2 points)**

Camera setup: Initially, the camera should look at the point (0,0,0) and be positioned at (0,30,25). The upward direction should be the usual (0,1,0) vector. **(2 points)**

1. Geometries **(total of 32 points, min. 15 points needed)**

1.1. Quad – square **(∑ 8 points)**

In the program, there is given a square in the XZ-plane with a unit side length ($x, z \in [-0.5, 0.5]$). This square is used for displaying the table and the cards.

1.1.1. **Table**. Size: 30 x 21, texture: `green_fabric.jpg`, position: 0.1 units below the origin. **(3 points)**

1.1.2. **Cards**. Size: 1.6 x 2.48, texture: `card_back.png` (in the shader part it will change!), position: always on the $y = 0$ plane. Initially 5 piece at points $z = 0$, $x = -4, -2, 0, 2, 4$. It is recommended to store the state of each card (position, and later card type as well) in a vector. **(5 points)**

1.2. Hand **(∑ 6 points)**

Load the hand model from the `hand.obj` file. Apply the `hand.png` texture to it. When rendering, scale the object to three times its original size **(2 points)**. Place the hand at the point $(-3, 2, 12)$ (the correct placement with the initial camera position is visible above) **(1 point)**. The hand should be rotatable around its own y-axis using `ImGui::SliderAngle` with a range from -45 to 45 degrees **(3 points)**.

1.3. Cylinder – Chip **(∑ 18 points)**

Load the poker chip object and apply the corresponding texture (`pokerchip.obj`, `pokerchip.jpg`). The base of the cylinder lies in the XZ-plane, with the center of the bottom face being the model's origin. The radius of the circle is 1 unit, and the height of the cylinder is 0.15 units.

Placements:

- N=10 chips stacked in a tower (directly on top of each other), with the bottom chip lying on the table at the point $(-10, 0, 5)$. **(5 points)**
- K=5 chips are rolling around the tower, spinning on their edges. The radius of the circle is 4 units, and the center is the tower. The chips are rolling, meaning they are also rotating during the movement. **(10 points)**
- The values of N and K should be adjustable using `ImGui::SliderInt` within a range from 1 to 20. **(3 points)**

2. Shader tasks **(total of 38 points, min. 15 points needed)**

Solution suggestion: Different objects may require different program sections on the shader side. To handle this, it is recommended to pass an integer value (int) that indicates which type of object is currently being drawn.

2.1. Swinging point light source **(∑ 18 points)**

In the scene, a white **point light source** should affect all objects according to the learned Phong model **(5 points)**. The light source's position follows the path $\boldsymbol{p}(t) = \begin{bmatrix} -15 + 30t \\ 10 - 20t + 20t^2 \\ 0 \end{bmatrix}$, where t changes continuously between 0 and 1, moving back and forth every 2 seconds. **(6 points)**

Let the quadratic attenuation coefficient be 0.001. **(1 point)**

Let the movement of the light source's position to be toggled on and off using `ImGui::Checkbox`. The light should continue from where it was last stopped (i.e. it should not "jump" when starting or stopping) **(3 points)**

An additional checkbox should toggle the lighting, allowing only the texture to be visible without shading. **(3 points)**

2.2. Chip texture **(∑ 10 points)**

In the shader, modify the texture color for the chips based on a uniform value. The 6 different colors can be achieved by swapping and rewriting the color coordinates. If the original color coordinates are (R,G,B), the six colors can be created as follows:

Red	Green	Blue	Yellow	Magenta	Cyan
(R,G,B)	(G,R,B)	(B,G,R)	(R,R,B)	(R,G,R)	(G,R,R)

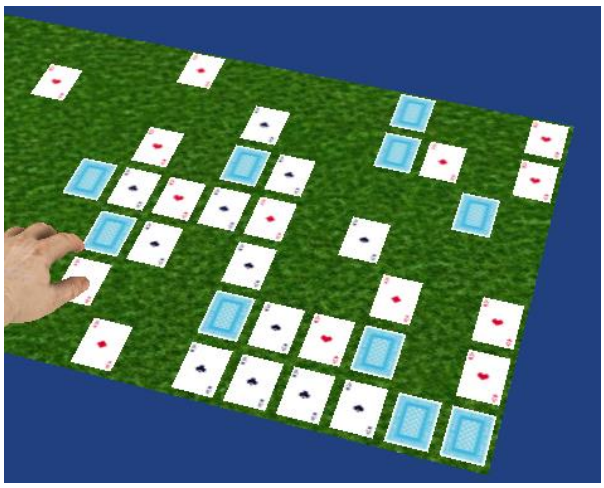
The chips in the tower should sequentially use the different colors; similarly, the rolling chips should also follow the same pattern. **(10 points)**

2.3. Card texture **(∑ 10 points)**

The different card faces are stored in a common texture (`cards.png`), which should be used for rendering the cards. To correctly display each card, the **u** texture coordinate in the shader should be divided by the number of cards (i.e. 5) and then add T / 5, where T is an integer value between 0 and 4 (indicating which card is being displayed). Initially, each card should appear once. **(10 points)**

3. Other tasks **(total of 40 points)**

3.1. Placing new cards on the table **(∑ 20 points)**



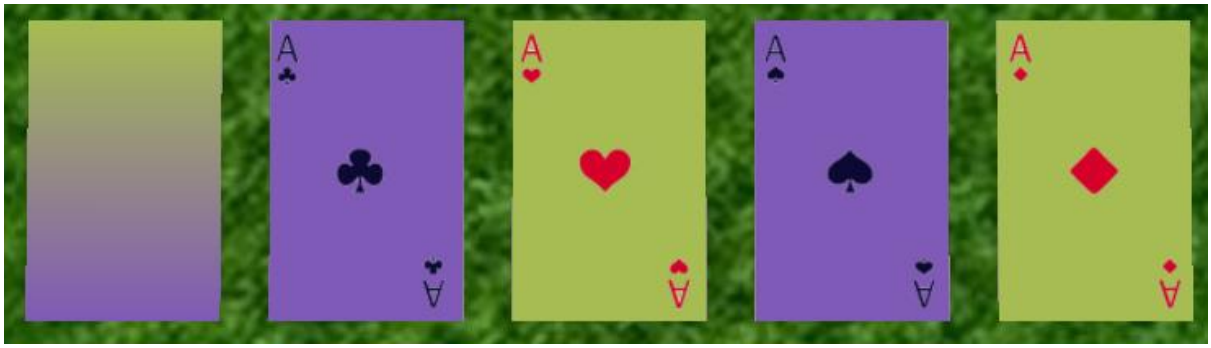
Allow the user to select a point on the table using Ctrl+Click (the interaction is already set up in the starter project, see Update()). Place a new card at the selected point (with $y = 0$ still). **(6 points)**

Allow the user to choose which type of card to place using the keyboard (see *2.3. Card texture* task). For example, pressing the keys 0, 1, 2, 3, 4 should switch the card type that will be placed. **(5 points)**

The placement should only work on the table's surface, and nothing should happen if the user clicks outside. **(3 points)**

The card placement should be restricted to a grid. When the user clicks on a point, find the nearest valid grid point to the clicked position (the card's center should be placed at (x,y=0,z), where x is an integer divisible by 2 and z is an integer divisible by 3). There's no need to check if a card already exists at that location. **(6 points)**

3.2. Swapping card texture **(∑ 20 points)**



In the GUI, allow the user to select two colors (`c1` and `c2`, `ImGui::ColorEdit3`).

When a GUI button (`ImGui::Button`) is pressed, create a new texture for the cards as follows. Load the original card texture (`cards.png`) into memory, and before passing it to the GPU, modify its contents as follows:

- The back side of the cards should be overwritten with a gradient (vertically between `c1` and `c2`) **(10 points)**
- For the black-suited cards (clubs, spades), the background (where its originally white) should be overwritten with color `c1` **(5 points)**
- For the red-suited cards (hearts, diamonds), the background (where its originally white) should be overwritten with color `c2` **(5 points)**

Some hints: The original texture is 2000px wide, and there are 5 cards on it, so each card occupies 400px. You can read and write pixels in the `ImageRGBA` structure (storing an image) using `GetTexel(x,y)` and `SetTexel(x,y,color)` functions, where `color` is represented as a `glm::u8vec4` value (RGBA color with each channel as an integer between 0-255). After the color modification, don't forget to delete the old texture and then pass the new texture to the GPU.