1. Practical Tasks: Basic Setup (1 Point)

Create a Kore application which displays a simple geometric form (for example a line or a rectangle). Make sure it actually works and push it to your team's Git repository. Start out by cloning https://github.com/KTXSoftware/Exercise1.git recursively (git clone --recursive).

You can find introductions to Kore and Git at http://wiki.ktxsoftware.com

Interesting solutions included:

- Use random numbers to create a random field
- Draw Shapes using lines
- Build a Mesh data structure and rotate the result ☺ Nice one, Group5!

The following code generates a nice gradient over the screen. (Credits to Group37)

```
for (int y = 0; y < height; y++)
    for (int x = 0; x < width; x++)
        setPixel(x, y, (float) x / width, (float) y / height, 0);
```

2. Theoretical Tasks: Light and Sound (5 Points)

2.1 Light Waves

List the basic wave parameters of electromagnetic waves in the visible spectrum (aka light).

Directions: transverse waves
- Propagation
- Oscillation

Amplitude
Speed: c = ~ 299 792 458 m / s
Wavelength: 400 – 700 nm

(Also naming the parameters without providing the concrete values was counted as correct)

2.2 Sound Waves

List the basic wave parameters of sound waves.

Directions: longitudinal
- Propagation
- Oscillation

Amplitude:
- Amplitude of the air pressure in the wave
- dB -> logarithm of the amplitude squared

Speed: ~ 340.29 m / s

Wavelength: 17 m - 17 mm

(Also naming the parameters without providing the concrete values was counted as correct)

2.3 Sci-Fi Monitors

What is the minimum amount of values necessary to encode a pixel for a bird who can see four base colors and the polarization of light?

Five values:
Four base colors -> 4 values
Polarization of light -> 1 value

2.4 Gamma Curves

Image data is usually gamma encoded. Which gamma value is optimally used for encoding for (a) internal lighting calculations and (b) for writing a final image to the framebuffer?

From the script: "Monitors usually operate using a gamma value of 2.2. Image data saved on a computer is generally encoded using an inverse gamma value of about 0.45 which cancels out the monitor gamma transformation such making the process of displaying an image simple again. Gamma encoded values are however not ideal for internal lighting calculations for games."

a) 1
b) 0.45. This value cancels out the monitor gamma of 2.2.

2.5 Vertical Synchronization

Consider a monitor that runs vsynced with 60 Hz and a game which consistently calculates one frame in 10 milliseconds. How much time does the game spend waiting for the monitor when (a) double buffering or (b) triple buffering is used?

a)
A frame can be rendered in 10 ms.
Every 16 ms the two buffers can be switched, then the back buffer becomes the front buffer. A waiting time occurs whenever the inactive back buffer contains a frame and the front buffer is still shown on screen.
The waiting times are about 6.66 ms.
The table uses a rounded value for the monitor synchronization.
Underlined times are the time steps.
Bold Frames are shown on screen.

Time	Active FB	FB 1	FB 2
0 ms	FB 2	Begin: F1	---
10 ms	FB 2	F1	---
16 ms	FB 1	**F1**	Begin: F2
20 ms	FB 1	**F1**	Active: F2
26 ms	FB 1	**F1**	F2
32 ms	FB 2	Begin: F3	**F2**
42 ms	FB 2	F3	**F2**
48 ms	FB 1	**F3**	Begin: F4
58 ms	FB 1	**F3**	F4
64 ms	FB 2	Begin: F5	F4

b)
Three buffers can be used to create a sequential buffer chain in which case the waiting times do not change compared to double buffering for constant rendering times (see buffertest.js). Chained triple buffering can only reduce render wait when frame times vary.
Three buffers can also be used run image rendering and image output completely independent of each other. Images are rendered to two buffers. The third buffer is only used to transfer image data to the monitor and does not directly interact with the application. Whenever the monitor can receive a new image, the last image that was rendered into one of the two application buffers is copied to the data transfer buffer. There are never any waiting times for the application when this setup is used, but the application eventually renders images which are never copied to the monitor, such as frame 2 in the following table.

Time	Active FB	FB 1	FB 2	FB 3
0 ms	FB 3	Begin: F1	---	**---**
10 ms	FB 3	F1	Begin: F2	---
16 ms	FB 1	**F1**	Active: F2	---
20 ms	FB 1	**F1**	F2	Begin: F3
30 ms	FB 1	**F1**	Begin: F4	F3
32 ms	FB 3	F1	Active: F4	**F3**
40 ms	FB 3	Begin: F5	F4	**F3**
48 ms	FB 2	Active: F5	**F4**	F3
50 ms	FB 2	F5	**F4**	Begin: F6
60 ms	FB 2	Begin: F7	**F4**	F6
64 ms	FB 3	Active: F7	F4	**F6**

Triple buffering makes it harder or even impossible for the application to calculate how long a frame is actually shown on the monitor. This introduces stuttering even though the framerate might be high.