Prof. Dr.-Ing. Ralf Steinmetz
Multimedia communications Lab

Dr.-Ing. Florian Mehm
Dipl.-Inform. Robert Konrad

Game Technology
Winter Semester 2016/2017

Exercise 2

For bonus points upload your solutions until **Saturday, November 12, 9:50**

General Information

- The exercises may be solved by teams of up to three people.
- The solutions have to be uploaded to the Git repositories assigned to the individual teams.
- **The submission date (for practical and theoretical tasks) is noted on top of each exercise sheet.**
- If you have questions about the exercises write a mail to game-technology@kom.tu-darmstadt.de or use the forum at https://www.fachschaft.informatik.tu-darmstadt.de/forum/viewforum.php?f=557

P2 Practical Task: Crack me up (5 Points)

Old games were often accompanied by little start animations called “cracktros”. Most cracktros are simply functional animations of positions and colors. YouTube contains hundreds of examples.

In this exercise, you can build upon the code you wrote for exercise 1 if you want to. Animate at least five properties of your graphics in a purely functional way. Example properties could be position, rotation, scale, colors, and many more.

Notes:

- If your texture(s) use PNG, drawTexture will use the alpha channel for alpha blending
- Unless you write your own code, there is no text rendering code in Kore

https://github.com/TUDGameTechnology/Exercise2.git contains additional code to help you out. You can either copy the code changes manually or just pull them into your own repository using
git pull https://github.com/TUDGameTechnology/Exercise2.git (and then handle any merges necessary).

Please remember to push into a branch called exercise2.

T2. Theoretical Tasks: Timing (5 Points)

T2.1 Iterative and functional time calculations (1 Point)

A space ship labeled Vic Viper starts at time t = 0 and position x = 10 with an x speed of 14 per second. Calculate the position x after 5 seconds (a) procedurally and (b) iteratively with 3 steps per second. (You can write a script or use a spreadsheet application).

T2.2 Different framerates (1 Point)

The situation starts out as described in 2.1 but there is a wall of width 5 at between positions x1= 60 and x2=65 (the space ship is a point, it has no width). Calculate the position of the space ship after 5 seconds for (a) a 3 steps per second and (b) a 2 steps per second update. Check for collisions in every update step. When a collision happens, move back the ship just so far that it barely touches the wall and set its speed to 0.

T2.3 Framerate (0.5 Points)

You are working on a game without using any form of multitasking. As it turns out, a purely graphical effect you implemented runs slow. A colleague suggests calculating and caching the effect every third frame only. Is this a proper optimization strategy? Discuss the resulting changes to the framerate and the game experience.

T2.4 Vertical Synchronization (1.5 Points)

Consider a monitor that runs with 60 Hz and a game that runs vsynced and which consistently takes 10 milliseconds to render one frame. How much time does the game spend waiting for the monitor (vsync) on average when

a) double buffering,
b) triple buffering with the mode introduced as FIFO,
c) triple buffering with the mode introduced as MAILBOX

is used?

Notes:

- Changing buffers is implemented using page flipping
- We assume frames are rendered in exactly 10 ms and page flipping is instant

T2.5 Performance (1 Point)

Consider the following parts of a program.

a) Where are the NPCs allocated: On the stack or on the heap?
b) What possible performance problem can you find in the code below? Provide an alternative that is more performant.

```
// Forward declaration, defined elsewhere
class NPC;


//...
// Initialization code
const int NumNPCs = 10000;
NPC** NPCs = new NPC*[NumNPCs];
for (int i = 0; i < NumNPCs; i++)
{
      NPCs[i] = new NPC();
}


//...
// Updating all NPCs
for (int i = 0; i < NumNPCs; i++)
{
      NPCs[i]->Update();
}
```