1. Practical Tasks: Crack me up (5 Points)

For reasons far beyond human comprehension 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.

Give your game the distinct TU Darmstadt style by adding a cracktro. Animate at least five properties of your graphics in a purely functional way.

https://github.com/KTXSoftware/Exercise2.git contains additional code to help you out. You can either copy

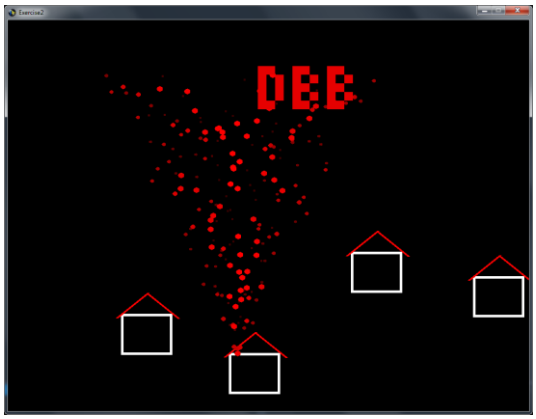
the code changes manually or just pull them into your own repository using git pull https://github.com/KTXSoftware/Exercise2.git

Thank you all for the nice solutions! Some of the most interesting ones used element such as…

- Animating sprites (e.g. draw one of two sprites based on the time)
- Shifting colors (e.g. using sin(t)*255 as input for r/g/b in drawPixel)
- Animating objects along a sine wave
- Segmenting an image and animating the segments/lines individually
- Animating objects on several levels (e.g. object follows a figure “8” and rotates at the same time)
- A tornado made of particles that rotate around a center axis, getting larger as they come towards the observer and smaller going backwards, and the group name being animated along with the particles.
- Text rendering based on an ASCII table
- Animating several objects on close trajectories
- Text with shifting colors animated on a sine curve
- Animating a particle star field that moves at different velocities and directions


The quick brown fox jumps over
Nintendo
Presents
super mario


BITSHIFT
mattes & randle >> GFX by alfredo, mattes & ra
L ROBERT


DBB


GAME TECHNOLOGY
SVEN PELDSZUS
OLF
DENNIS W
ATHIONA XHOGA
TECHNISCHE
UNIVERSITÄT
DARMSTADT


Everglade
Studios


TU Darmstadt
GRUPPE 5


Exercise2
FOO BAR CRACK
BuildingBlockGames
NEW GAME

2. Theoretical Tasks: Timing (5 Points)

2.1 Iterative and functional time calculations

A space ship labeled Vic Viper starts at time 0 and x position 10 with an x speed of 14 per second. Calculate the x position after 2 seconds (a) functionally and (b) iteratively with 3 steps per second.

a)

x = 10 + 2 * 14 = 38

b)

deltaX = 14/3 =~ 4.67

x = 10 + (3*2) * deltaX = 38.02

Note that in this case, the result is slightly different due to rounding errors. If you worked with fractions and got an exact result of 38, all points were still given.

2.2 Different framerates

The situation starts out as described in 2.1 but there is a wall of width 5 at x position 60 (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.

Timestep	deltaT = 14/2	Time	deltaX = 14/3	Time
0	10	0	10	0
1	17	0,5	14,66666667	0,33333333
2	24	1	19,33333333	0,66666667
3	31	1,5	24	1
4	38	2	28,66666667	1,33333333
5	45	2,5	33,33333333	1,66666667
6	52	3	38	2
7	59	3,5	42,66666667	2,33333333
8	66	4	47,33333333	2,66666667
9	73	4,5	52	3
10	80	5	56,66666667	3,33333333
11			61,33333333	3,66666667
12			66	4
13			70,66666667	4,33333333
14			75,33333333	4,66666667
15			80	5

The table above shows how the situation would look in case no collisions were handled.

a)

We see that there the Vic Viper is inside the wall at time 2.67. We reset it to the position of the wall x= 60 and it has no more speed, so it stays at x=60.

b)

At t = 3.5, the ship is infront of the wall, at t = 4, it is clear of the wall again. If we do not check for collisions in between, tunneling has occurred: a small moving object has passed through a relatively thin collider between our calculations.

One way of finding this collision would be to check a line between the positions at t=3.5 and t=4. If the line collides with something, we can find the time when the first collision happened and handle it. However, this might be inaccurate. Better would be to cast the shape of the space ship along the line of movement, which is more costly to calculate, however.

2.3 Framerate optimizations

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

The net framerate would increase. However, the frame rate would not be constant. For every two “quick” frames, one “slow” frame (in which the effect is calculated) would happen. An inconsistent framerate like this usually looks more choppy than an overall lower but more consistent framerate.

This was the main answer that was expected in this task. A few groups pointed out that the effect might look bad at a third of the frame rate, for example if it was an explosion that was running at a lower frame rate as the rest of the game.