

Status

In Ubec currently all five games/puzzles are implemented. The only difference with the concept is a slight change of the gamelogic in some parts (no crashing object, eg.). Where the effects can be found is described in the following section.

Controls

World movement

- hold and drag right-mouse-button for orientation
- use scroll wheel for zoom
- F1: controls; F5: wireframe mode; F6: backface culling; F7: show “debug” game-info

Game interaction

BallGame – use arrow keys to set the ball in motion (view-depending)

MovingTiles – left-click on a tile, to swap it with the blank space

RiverCrossing – left click on: Entity to move onto the boat, Boat to move Entity back on bank, River left/right to the boat to move the boat to the opposite bank.

RedButton (CubeBottom) – left click to push button.

FlowPuzzle – left click on field- color changes with the amount of clicks

Playthrough

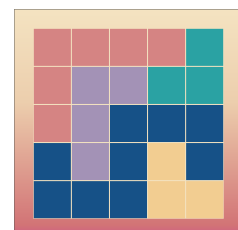
BallGame – Goal of this puzzle is to find the right hole for the ball. Look at the game from the bird’s-eye-view. Your target hole is on the bottom right of the surface, diagonal from your starting position. Skillfully try to roll in the ball and solve the puzzle.

MovingTiles – Rotate the four tiles in the top left clockwise to complete the picture (last pile appears in top-left position, when other tiles are correct)

RiverCrossing – One possible solution [1]

- put deer on the boat and transfer to other bank, go back empty
- put broccoli onto boat, transfer and load the deer for the back travel
- unload deer and put dog, transfer, go back empty
- put deer onto boat and transfer, puzzle should be solved

FlowPuzzle- Connect all color origins with the according color:



Press the same field multiple times to change the color and connect all the paths.

RedButton - Press the button when all other games are solved to stop the time and finish the game.

Gameplay

3D Geometry (6 Points)

Models (inkl. bones and animations) can be imported via assimp library. Shown in e.g. RiverCrossingGame.

Playable (3 Points)

Shown in all games.

Advanced Gameplay (3 Points)

BallGame - You can roll a ball through a labyrinth, which consists of platforms, walls, ramps and holes. The goal is to find the right hole, drop the ball and win the game. If you choose the wrong opening or the ball drops off the edge of the cube, the game will reset and you get a new try.

Min. 60 FPS and Framerate Independence (3 Points)

FPS currently not a problem- besides that instanced rendering is used, to ensure min FPS in the future. Furthermore the physics will be sampled and updated at 60 FPS to ensure the same speed and sensitivity on all machines.

FI shown in BallGame, paricleSystems and animation

Win/Lose Condition (3 Points)

The game is won, when all puzzles are solved.

Lose, when the time is up.

In either case the scene is changed to a simple message.

Intuitive Controls (2 Points)

BallGame - If one of the arrow-keys is pressed once, the Ball will roll in the matching direction (view-depending). It will not continuously roll, instead it slowly stops after a while and you will need to push again.

Everything else is controlled by the mouse- interaction with left and rotation with right click

Intuitive Camera (2 Points)

Movement is fixed around the cube, which is the center of attention. Scrolling is possible. Other movements don't really make sense in our game.

Illumination Model (2 Points)

One directional light is used (best seen on BallGame with shadows)

Parameters can be set at object creation, normals either imported or generated.

Textures (2 Points)

Are used throughout the whole game.

Moving Objects (2 Points)

BallGame - the ball can be set in motion by pressing the arrow-keys.

MovingTiles- the tiles can be set in motion by left clicking a moveable tile.

Collision Detection (Basic Physics) (4 Points)

Using PhysX Library.

BallGame - Physics are used to create a dynamic Ball and a static Labyrinth. The ball can be set in motion via adding Force and won't go through walls.

MovingTiles/RiverCrossingGame- Physics are used to interact with the game. Static objects represent hitboxes for an interaction ray (sent from the camera's position into the view direction). When such a ray intersects with a hitbox, an action is set.

Heads-Up Display (4 Points)

So far the text is rendered as a rectangle with glyph textures (and transparent background). View and projection are "ignored" to "stick" the rectangles onto the screen.

Currently shown all the time (FPS and timer in corners of the screen), toggleable with F7. Currently "text" is rendered, using FreeType-Library. Image/Controls are shown when pressing F1.

Effekte

Lighting

Shadow Map with PCF (16 Points)

A ShadowMap is rendered from the lights perspective (only a depth map is generated), which is the base of the following shadow calculation. The shadow map is rendered with a FBO. Artefacts like shadow acne and peter panning are avoided by using backface culling- this is possible because we only use objects with thickness in our scene. Jaggy edges are avoided with PCF on the hardware and combining 16 different, poisson sampled, points in the transition area. For performance reasons, only four are used in homogenous areas (baily early). Can be seen in BallGame.

Advanced Modelling

Particle System

Particles in our case are points, where a certain texture is stitched to it. The normal vector of these planes always faces to the user. New particles are generated, when old die and the color, size and position change over the adaptive lifetime. Well shown in FlowGame.

Animation

GPU Vertex Skinning (20 Points)

Modes are imported with bones and animation. The vertices are transformed and interpolated according to the weight of the bones and keyframes. The weight and the matching Bone is than passed to the modelLoadingShader. Shown in RiverCrossingGame by clicking one of the animals.

Texturing

Environment Map (8 Points)

A cubemap is used as a source of reflection for the environment mapped object. The reflection combined with the material properties of the object, give the look of the object. Shown on the base cube/frame. Additionally a skybox was used to give the game a specific mood.

Cel Shading (4 Points)

The surface brightness gets divided into levels. Each color will be set to the lower limit on their level. The results are discrete color layers. Shown in RiverCrossingGame and CubeBotton (Button on bottom of cube).

Post Processing

Contours via Backfaces (4 Points)

Contours via EdgeDetection was switched to Backface, due to unwanted functionality of EdgeDetection.

Backfaces are drawn slightly larger in a contour texture before front faces are drawn.

Currently used on the Ball from BallGame to slightly separate movable objects from the static environment. Shown in RiverCrossingGame (Clickable Models) and BallGame (movable Ball).

Libraries und Code Sources

“The Open-Asset-Importer-Lib” for model import (<https://www.assimp.org/>)

“The FreeType Project” for rendering 2D-text (<https://www.freetype.org/>)

“NVIDIA PhysX SDK 4.1” to add physics (<https://github.com/NVIDIAGameWorks/PhysX>)

Slightly modified source code for mouse picking

(https://github.com/andersonfreitas/opengl-tutorial-org/tree/master/misc05_picking)

Print text on screen (<https://learnopengl.com/In-Practice/Text-Rendering>)

[1] <https://de.wikipedia.org/wiki/Fluss%C3%BCberquerungs%C3%A4tzel>