

Documentation

Group/Game Name: Magnets

Brief description of implementation: The player must cross treacherous obstacles on their way to solve the mysterious puzzles facing them, all using the powers of magnetism.

Features:

The player can give objects a magnetic charge, leading them to attract or repel each other accordingly. The player can lose by falling into pits of ominous goo and win by making it to the other side of the rooms, through a total of five levels.

All other features are better described by their associated effects.

Additional libraries:

Physics and collision handling: PhysX

(<https://nvidia-omniverse.github.io/PhysX/physx/5.6.1/index.html>)

Window handling: GLFW (<https://github.com/glfw/glfw>)

Math: GLM (<https://github.com/g-truc/glm>)

JSON handling for scenes and options loading: JSON (<https://github.com/nlohmann/json>)

PNG loading: LodePNG (<https://github.com/lvandeve/lodepng>)

GLTF loading: TinyGLTF (<https://github.com/syoyo/tinygltf>)

Gameplay:

Mandatory:

- 3D Geometry: GLTF models from the `assets/models` folder are being loaded using the TinyGLTF library. All models were made by ourselves in Blender.
- Playable: The player can move and look around and climb small ledges. The player can also interact with cubes and magnetizable objects in the scene.
- Min 60 FPS and Framerate Independence: Movement and physics simulation speeds are kept independent of the framerate by scaling with the time between frames (`delta_time`). We're running on a solid 60+ FPS on our non-high end hardware. The framerate is displayed in the window title.
- Win/Lose Condition:
 - Win condition: make it to the other side of the room.
 - Lose condition: fall into a pit of goo along the way.
- Intuitive controls:
 - The game uses standard WASD input for movement. Jumping, crouching and sprinting have not been included by design. The player can climb small ledges by walking into them.

- Magnetism mechanics use the mouse buttons: left mouse for positive charge, right mouse for negative charge, middle mouse (mouse wheel) for removing charge.
- Intuitive Camera: standard mouse controls
- Illumination model: Phong
- Textures: All visible objects have a texture. Textures are either loaded from files or (in the case of the goo) generated on the fly.
- Moving Objects: Central to gameplay, with the magnetism mechanics.
- Documentation: You're reading it.
- Adjustable Parameters: Screen resolution, fullscreen mode and a bunch of other options can be set in the `assets/options.json` file.

Optional:

- Advanced Gameplay: The player must navigate five different rooms featuring cube and button-based magnetism puzzles.
- Collision Detection (Basic Physics): Backed by PhysX, our scene contains both dynamic and static RigidBody objects, which collide with each other. The player also collides with physics objects and is not able to pass through them.
- Advanced Physics:
 - The magnetism mechanic requires the targeting of objects (raycasts) and comparatively complex force calculation modelled after gravity simulations.
 - Specifically, for all magnetized objects it must be calculated whether they attract or repel, before calculating the correct force vector and applying it to the object, assuming it is a dynamic one.
- Scripting Language Integration: None
- View-Frustum Culling: Implemented using a kd-tree, see below.
- Heads-up Display: a dynamically updating reticle that shows how many objects in the scene currently hold a magnetic charge.

Effects:

Lighting:

- none

Advanced Modelling:

- none

Terrain:

- none

Animation:

- Vertex Shader Animation:
 - The goo at the bottom of the pit is animated using a vertex shader.
 - The shader uses the simplex noise algorithm implementation by McEwan and Arts, as provided by <https://thebookofshaders.com/11/>.
 - Roughly speaking, the y coordinate of the vertex is displaced by an amount dictated by the noise at that location (adjusted by a subtraction of 0.5 to make sure the vertices are lowered for values below 0.5 and multiplied by another

constant to scale the strength of the displacement). The offset is increased and decreased over time, as dictated by a sinus function (which is also offset using noise and its speed adjusted with a constant).

- Normals are recalculated by two points offset along each the x and z axis, before calculating the vectors between them and taking the cross product
- If difficult to see, we recommend watching the edges of the goo.

Texturing:

- Procedural Texture:
 - The "goo" fragment shader generates a procedural texture for the goo.
 - It uses the same noise algorithm as mentioned in the section on Vertex Shader Animation.
 - Once again, roughly speaking it uses the noise to mix a brighter colour with its base. This leads to (moving, since the sampled coordinates are offset by time) yellowish spots on the green goo. To add darker areas, the dot product is used to compare the normal vector with the y axis, with more upright normals resulting in a higher contribution of a darker colour (via the mix function).

Shading:

- none

Advanced Data Structures:

- kd-Tree:
 - Used for culling in the scene. The amount of objects culled this way is displayed in the window title and it can be enabled or disabled by pressing F8.
 - The kd-tree is (currently) limited to culling static rigidbody objects, as continuously updating the position of dynamic objects could easily lead to a net loss of performance in our case.
 - The tree is made up of nodes, with each node storing which axis it is splitting the tree on (as determined by its parent, a child of an X-splitting node will always split on Y and so on) and the value it is splitting on. Any object that intersects this plane is stored in the node, if not, it is passed to the left or right child, depending on which side it falls on.
 - To calculate visibility, the axis aligned bounding box of the view frustum is passed to the tree every frame, with all objects intersecting the frustum having their visibility set to true.
 - Splitting planes can be displayed by pressing F6, the size of a child node's plane is $\frac{1}{2}$ that of its parent.

Post Processing:

- Contours via Edge Detection
 - After falling into the goo, the player's field of vision begins to spiral and turn green, while surrounding edges are highlighted.

- Detection is done using the Sobel operator.

Other special features:

Point and directional lights

JSON Scenes:

Scenes and all their contents - objects, rigid bodies, materials, HUD elements, etc. - are loaded from JSON files in the assets/scenes folder. To that end, there's also a "type registry" system for objects and rigid bodies.

Post-processing Effects:

A list of our post-processing effects, in the order they are cycled through by pressing F9 (press F10 to return to default).

- Goo-Death
 - default, as described in "Contours via Edge Detection"
- Contours
 - adds contours to detected edges
- Sobel
 - the output of the Sobel operator after a threshold is applied
- Pixelation
 - calculates the average color in a particular section of the screen
- Chromatic Aberration
 - shifts where particular colors are sampled

Walkthrough:

Level 1:

Magnetize the cube with an arbitrary charge and the wall in front of you with the opposing one. Walk up the stairs (and cube) to reach the end of the room.

Level 2:

Magnetize the cube one of the walls on the sides of the pit with opposing charges, the cube should fall in and form a bridge across the goo. Cross it to reach the end of the room.

Do not fall into the goo. The goo will kill you.

Level 3:

Magnetize the cube (+) and surface (-) on the ceiling, the cube should float up to it. Magnetize the surface at the top of the far wall (-) and remove magnetization from the one on the ceiling. Remove the magnetization of the surface on the far wall. Walk up the stairs and magnetize one of the surfaces on the side of the pit (-) to your right. Use the cube to cross the pit and reach the end of the room.

Do not fall into the goo here either. The goo will still kill you.

Level 4:

This level introduces buttons. Magnetize the far side wall of the pit (+) and one of the two cubes (-). Demagnetize the pit wall. Magnetize the other cube (+), making it "topple" over the first

cube, into the pit. Play around with magnetization and demagnetization to make the two cubes form a bridge. Demagnetize both cubes.

Walk across the pit. Magnetize one cube (+) and the ceiling surface (-) to make the cube float upwards. Magnetize the surface near the button (-), then demagnetize the ceiling; the cube should end up hovering above the button. Demagnetize the cube and/or the wall. Once the cube lands on the button, the wall disappears. Walk to the end of the room to win.

Yet again, do not fall in the goo. It will still be harmful to your general condition.

Level 5:

Magnetize the cube (+) and the surface next to it (-). Remove the magnetization from the surface and let the cube drop onto the button. Walk across the newly-appeared bridge to the surface in the center. Turn around and magnetize the surface on the left wall (-) to remove the cube from the button. Turn around again and walk across the bridge to the end of the room.

You probably know about the goo by now.