apartment 42

implementations

gameplay

the goal of this version of the 'apartment 42' room-escape game is to find a way out by activating the door in room 1.
in order to accomplish this, you have to first find the (remote) key for the door.

you can interact with objects whenever the crosshair changes from a big cross to a smaller cross. optionally a beep could be set as an aditional feedback (see more on that in the features section).

complex objects

i'm using assimp for importing (collada) model files generated with sketchup.
in this version of my game i didn't want to ship the actual 'apartment 42' model (for various reasons, but not because of performance issues!). i just specially designed a simple two-room model and created a few riddles to show the gameplay, the way objects are animated and how the user can walk through the game.
the final version of the game (the one i'll present at the second game event) will have the "real" apartment model though.

animated objects

some objects are interactive (they have actions/animations applied). to interact with such an object you have to click the object.
an action can have several preconditional actions and postconditional actions applied.
to successfully start an action, all of its preconditional actions have to be active.
after the successfull start of an action all of its postconditional actions are (tried to) be started.
actions can happen instantly or with a specific progression time. they can have an automatic reset time set or they have to be clicked (a second time) to be reset.
during an animation the direction of progress can be reversed by simply clicking the object.

view-frustum culling

i implemented view-frustum culling with bounding spheres. this feature can be toggled with the '8' key.

transparency

transparency can be seen in the action feedback notification (the notification which appears when the user clicks an animated object) as well in the debug info. the action feedback notification will automatically fade out after a specific time.
there are no transparent objects.

experimenting with openGL

i'm using VBOs, VAOs. for deferred rendering i'm using an FBO (with a separate color, normals and a depth texture).

key mappings

2 - debug info (framerate, rendered triangles count, state of view-frustum culling and gravity)
3 - switch render mode: wireframe / fill mode)
8 - toggle view-frustum culling

I - show/hide inventory
M - release/fetch mous pointer
P - toggle post-processing filter
G - toggle gravity
W/S, A/D or arrow keys - walk

ESC - close the game

left mouse button - activate an interactive object

effects

several post-processing filters:
* edge detection with original image
* cartoon shading
* CAD style (black lines on white background and vice versa)
* depth of field (with a 3x3 and a 5x5 gaussian filter)

the post-processing filter can be choosen and set in the settings.xml file (more on that below).

special features

settings.xml and actions.xml

for flexible settings and actions management i implemented a simple XML reader (based on http://pugixml.org/).

with the two files one can easily customize (important) settings of the game as well as add/delete/edit actions.
the actions.xml file has a special function as it's the "place" where plain objects of the model become interactive objects. it is basically a finite state machine on which the whole game (and all its riddles) is based on.

what can be set in the settings.xml file:
* keymappings
* game title and introduction message (goal, ...)
* window size
* type of post-processing filter
* the actual model file
* various camera settings:
  - position,
  - looking target,
  - collision detection radius,
  - eye height,
  - walking speed, ...
* UI settings as
  - the font (name and color) to be used (action feedback, debug info, inventory, ...),
  - times for the action feedback notification (how long it will be shown and the fade out time)
  - crosshair settings (size, line width, beep as the aditional feedback, color)
* settings for the outlines (line color and width)

how does an action look like:
* id: number that gets referenced by pre/post-conditions
* node: to which node of the model the action should be applied
* description: textual description of the action
  (acts also as the feedback when an action could not be run because one of its preconditions are not fulfilled)
* transformation matrix (translation, rotation, scaling)
* times: how long it should take (forward and backward progress), auto reset time
* preconditions: list of actions (a_id references the action id) and in which state the different actions must be
  there is also the possibility to reset all preconditions if the action is tried to be run but at least one precondition is not fulfilled
* postconditions: list of actions (a_id) that are tried to run after successfully starting of this action

collision detection

collision detected is implemented as a sphere-sphere-intersection test.
the camera has a specific position and a given bounding sphere radius. the actual sphere center of the camera is the position (eye height) minus two times the radius of the sphere. with an eye height of 1.70 m and a radius of 30 cm the user can walk below objects with a clearance height of at least 1.70 m and "climb" objects (or walk stairs) which are at maximum 90 cm height.
collision detection is applied dynamically, meaning that if an object is animated also its bounding sphere changes and further a collision could or could not occure during different states of animation.

to dynamically adapt the camera position (to keep the given eye height) i just render the scene with adapted camera and view-frustum settings: camera down, very small field of view angle, view-frustum set to only include objects within -h and +h (h being the hight of the highest object to climb) and then read the center value of the depth buffer (texture).

for the collision detection i do not use any libraries or code samples. everything is done with pure math (which, by the way, i love!).

walkthrough

just explore the two rooms.
for the elevator riddle: there is a hint somewhere in room 1.
also, it's important to read the feedback when trying an action. it will tell you the first action (of the list of preconditions) that is not fulfilled yet.