EDAF80 Introduction to Computer Graphics

Seminar 5

End Game

Pierre Moreau

2019 Slides by Carl Johan Gribel, 2010-13, and Michael Doggett, 2014-17

Announcements

- If you have a disability, you can ask for help for the exam (extra time, or writing on a computer for example):
    1. Contact Christina Rowa, accessibility officer for LTH: https://www.lunduniversity.lu.se/student-life/preparing-to-come/students-with-disabilities
    2. Contact Michael Doggett, with the attestation given by the accessibility officer, **before the 14th of October**; the help is subject to Michael's approval.
- Register for the exam in LADOK
    - If you cannot find the course in LADOK, send me an email.
- Possible extra lab session in week 8 (21st Oct. to 26th Oct.)
    - Check the forum for more info; will also be announced during the lectures.

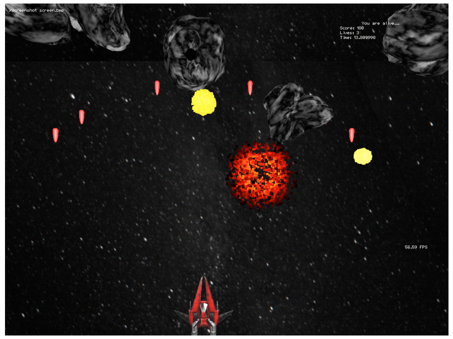
Today

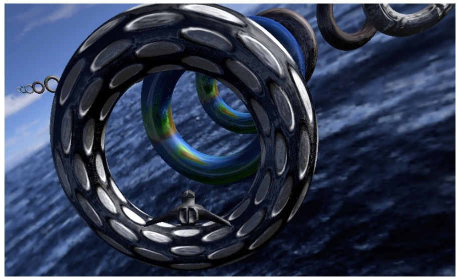
- Final assignment: **make a game**
  - Some ideas
- New stuff
  - Collision detection
  - Physics (inertia)
- Miscellaneous helpers
  - Add new files to the project
  - Load an external 3D model
  - Share your games with others

Game ideas

- **Asteroids**
  - shoot asteroids randomly towards camera/spaceship

  objective: Avoid and/or shoot them down





- **Torus ride**
  - place torus "rings" along path

  objective: Gain points by flying through
- ***Your own idea***
  - *consult us*

General considerations

- Fixed or POV camera?
- Manoeuvre by keys (WASD), mouse, both?
  - constrained to a plane, or full 3D?
- Animations: fixed, random, interpolation ..

Asteroids

- Fixed array of asteroids

  ```
  Node asteroids[N];
  ```

  - respawn when behind camera or shot down
  - hide/unhide:

    ```
    if(visible) asteroids[N].render(…);
    ```

- Randomize position, velocity vector etc
- Alter appearances using size, shaders, tessellation, noise …

Torus Ride

- Fixed array of tori

```
Node tori[N];
```

  - Fixed or infinite (respawn) path
  - hide/unhide:

    ```
    if(visible) tori[N].render(…);
    ```
- Place tori e.g. along random spline
- Alter appearances using size, rotation (spin?), shaders, tessellation …

When you're done...

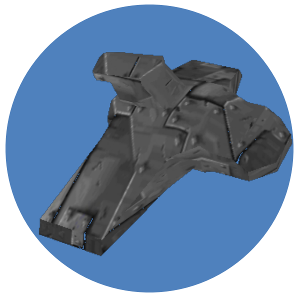
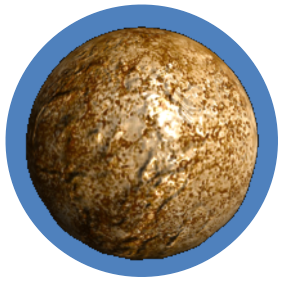
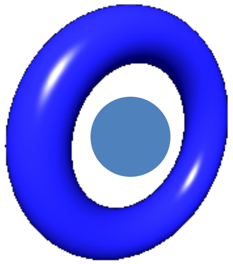
- Make a short post on the course forum presenting your game
  - Title and game objectives
  - Creators
  - Features and how you implemented them
  - Screenshots
- Present in Lab 5, week 7

Today

- Final assignment: **make a game** ✔
  - Some ideas
- New stuff
  - Collision detection
  - Physics (inertia)
- Miscellaneous helpers
  - Add new files to the project
  - Load an external 3D model
  - Share your games with others

Collision detection

- Use **bounding spheres** (BS) and perform *sphere* - *sphere* or *ray* - *sphere* collision tests
  - Cheap tests
  - Avoid other primitives

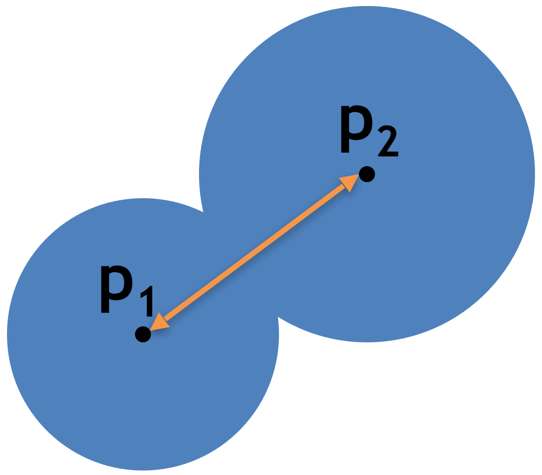




- **Note:** no need to use an actual *sphere* – just *position* + *radius*!
- List of intersection tests between many different objects

Sphere - Sphere

- Intersection if

$$| \mathbf{p}_1 - \mathbf{p}_2 | < r_1 + r_2$$



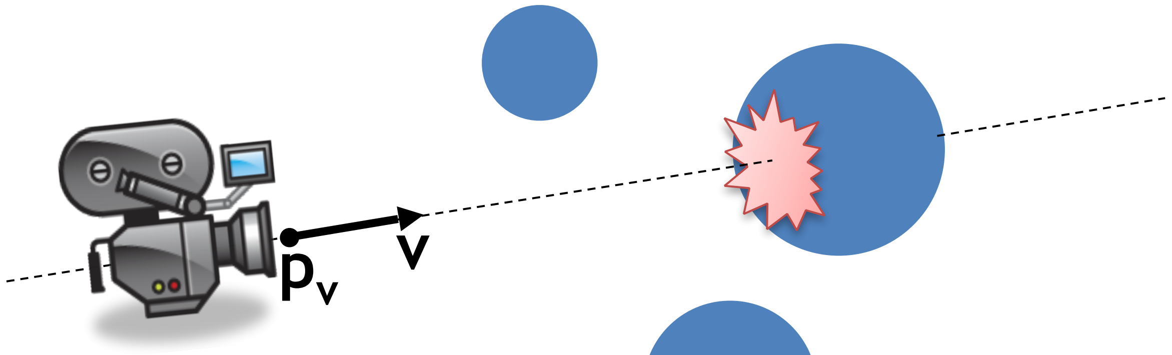
```
bool testSphereSphere(p1, r1, p2, r2)
```

Ray shooting

- Ray origin $\mathbf{p}_v$, unit direction $\mathbf{v}$

- ”Shoot” ray from camera

```
pv = mCamera.mWorld.GetTranslation();
 v = mCamera.mWorld.GetFront();
```



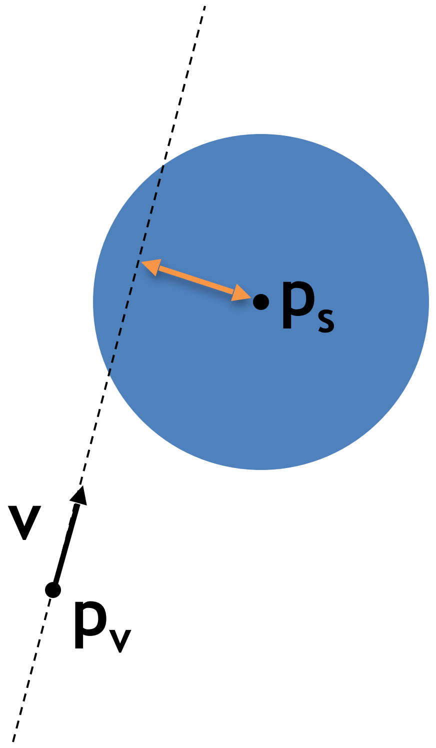
Ray - Sphere

- Ray origin $\mathbf{p}_v$, unit direction $\mathbf{v}$
- Sphere at $\mathbf{p}_s$, radius r

- Intersection if

$$\left|\ \text{rejection}(\ \mathbf{p}_s - \mathbf{p}_v,\ \mathbf{v}\ )\ \right| < r$$

$\text{rejection}(\ \mathbf{u},\ \mathbf{v}\ ) = \mathbf{u} - \mathbf{v}(\mathbf{u}\cdot\mathbf{v})$



```
bool testRaySphere(pv, v, ps, r)
```

Collision detection: Spaceship & asteroids sketch

- Spaceship and its BS radius:

```
Node ship
float ship_BSradius
```

- Asteroid & radii lists:

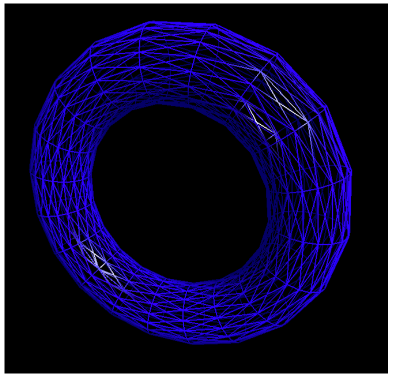
```
Node asteroids[N]
float asteroid_BSradii[N]
```

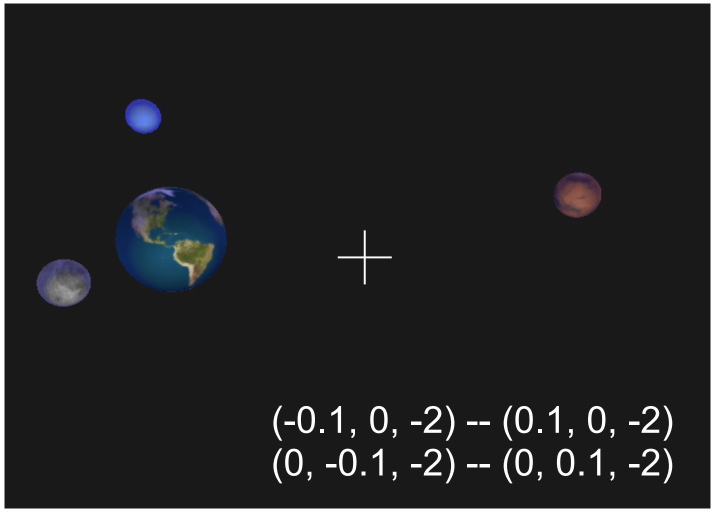
- Each frame, test spaceship against all asteroids:

```
for (int i=0;i<N;i++)
{
    if testSphereSphere(
                    worldPosition(ship),
                    ship_BSradius,
                    worldPosition(asteroids[i]),
                    asteroid_BSradii[i])
          { // lose life/award point… }
}
```

Drawing lines

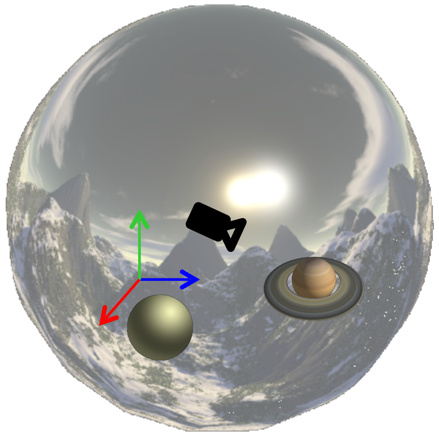
- Create a vertex array of line segments and set **mesh_data::drawing_mode** to **GL_LINES**
  - Then fill in the vertex array and use it as in assignment 2
  - i.e. same code as for parametric shapes, with the addition of changing the **drawing_mode** and creating lines rather than triangles.
  - Apply shader etc as usual
- Line width
  - Call **`glLineWidth($width)`** (glLineWidth documentation)
- Crosshair, "laser" etc
  - attach node to camera





Cube-mapped background

- Big sphere
  - Position around the scene, or camera
- Apply cube mapping shader
  - Sample the cube map using sphere’s world space **normal** instead of **reflection**
  - Disable culling:
    `glDisable(GL_CULL_FACE);`



Physics: acceleration/inertia

- Use fixed **acceleration** instead of fixed **velocity**
  - smooth starts and stops
- Sketch:

  init:

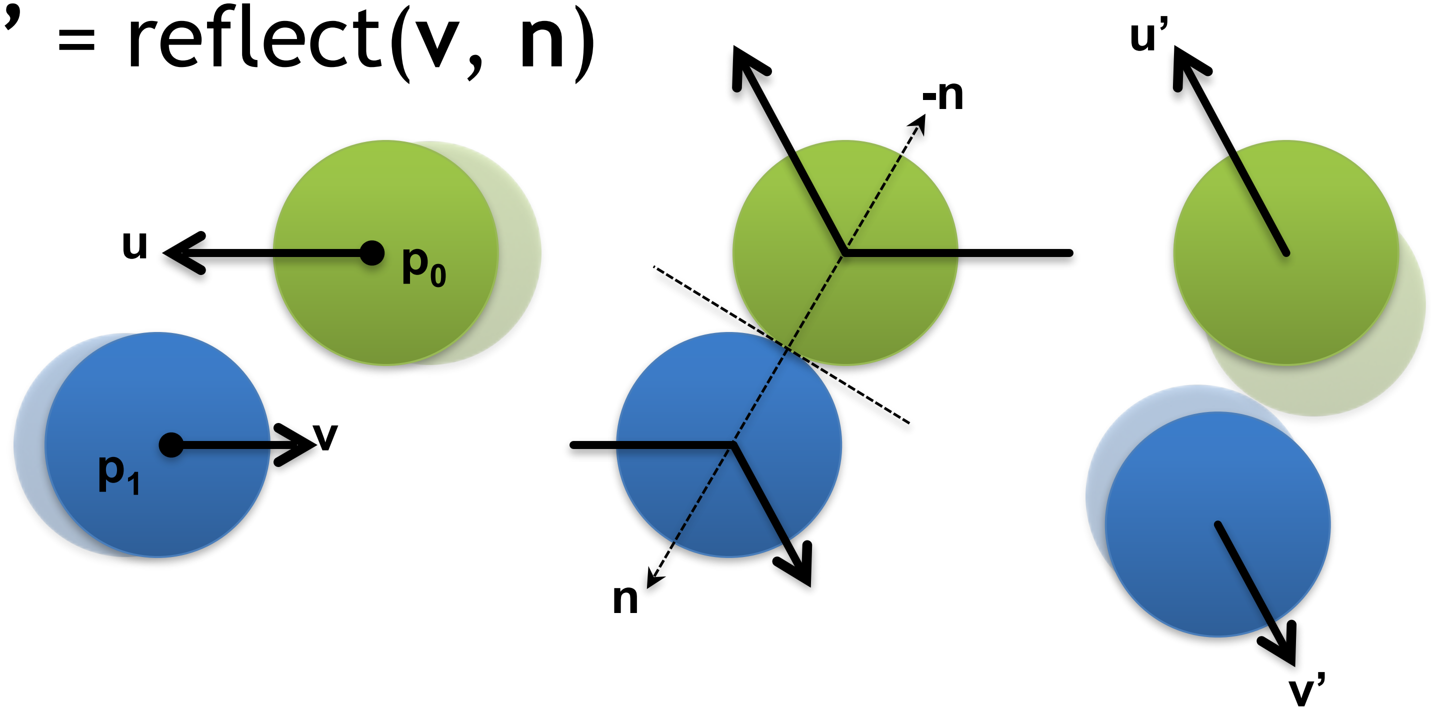
  ```
  glm::vec3 v = (0,0,0);
  ```

  each frame:
  compute `move` and `strafe` as before

  ```
  v = v + mCamera.mWorld.GetFront() * move
        + mCamera.mWorld.GetRight() * strafe;
  mCamera.mWorld.Translate(v*dt);
    // dt is time delta in camera update() function
  ```
- The trick: Euler integration of Newtons second law, *F=ma*

Physics: elastic collision

- Reflect trajectories along collision normal
  n = normalize($\mathbf{p}_1$-$\mathbf{p}_0$)
- **u'** = reflect(**u**, -**n**)
  v' = reflect(**v**, **n**)

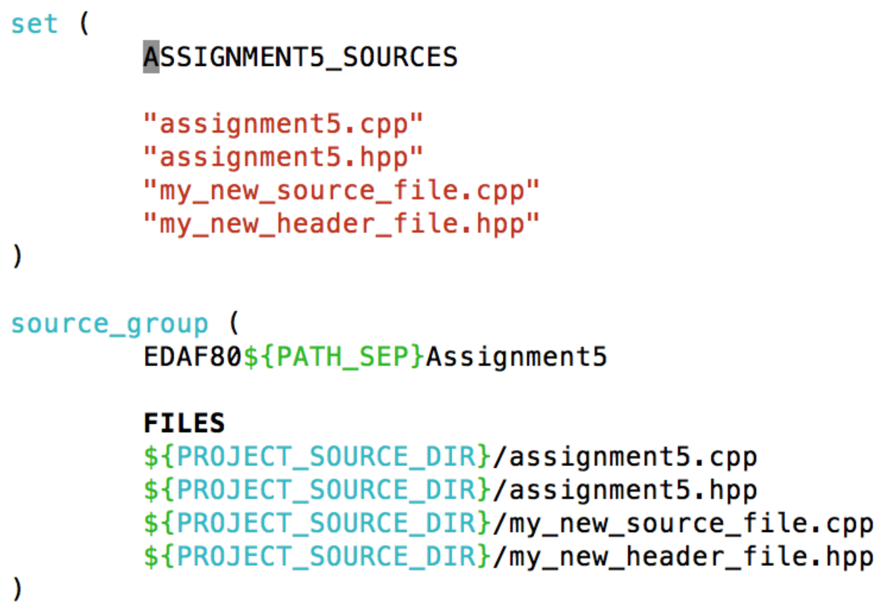


Today

- Final assignment: **make a game** ✓
  - Some ideas
- New stuff ✓
  - Collision detection
  - Physics (inertia)
- Miscellaneous helpers
  - Add new files to the project
  - Load an external 3D model
  - Share your games with others

Adding new files to the project

- Open **src/EDAF80/CMakeLists.txt**
- Append the files names to the variables **ASSIGNMENT5_SOURCES** and **EDAF80${PATH_SEP}Assignment5**
- For example, two new files added: *my_new_source_file.cpp* and *my_new_header_file.hpp*



```
set (
        ASSIGNMENT5_SOURCES

        "assignment5.cpp"
        "assignment5.hpp"
        "my_new_source_file.cpp"
        "my_new_header_file.hpp"
)

source_group (
        EDAF80${PATH_SEP}Assignment5

        FILES
        ${PROJECT_SOURCE_DIR}/assignment5.cpp
        ${PROJECT_SOURCE_DIR}/assignment5.hpp
        ${PROJECT_SOURCE_DIR}/my_new_source_file.cpp
        ${PROJECT_SOURCE_DIR}/my_new_header_file.hpp
)
```

Adding new files to the project

- If using Visual Studio 2017 (and built-in CMake support): create the files directly from Visual Studio.
- Otherwise: create the files manually, in the same folder as the other assignment files.
- Just build the project and start using those new files.

Load an external 3D model

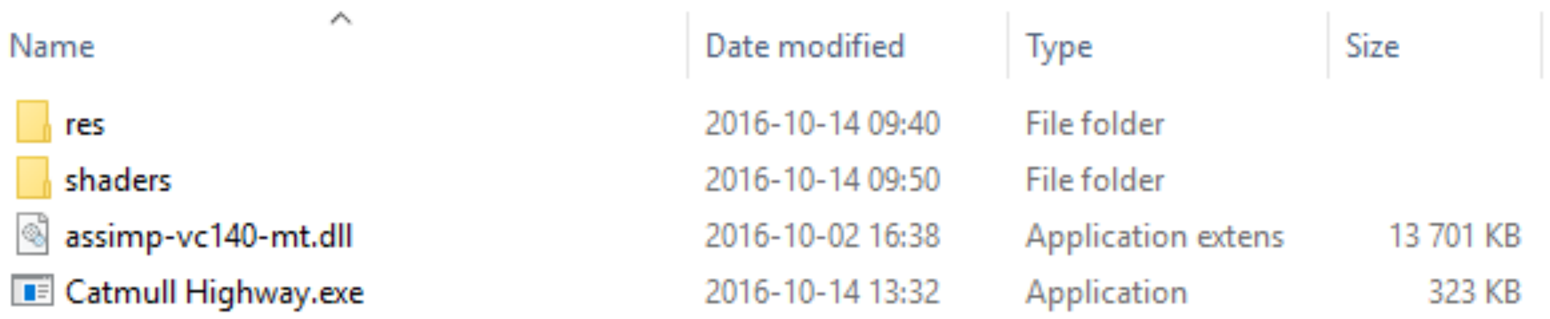
- Look at **src/EDAF80/assignment1.cpp**: the sphere for the planet was loaded that way!
- Use the **bonobo::loadObjects(filename)** function, from **src/core/helpers.hpp** **filename** is specified relative to **res/scenes** folder
- Returns a vector of **bonobo::mesh_data**, whereas **createSphere()** and the others only returned one instance of **mesh_data**

Share your game!

- Copy in a given folder, the following:
  - the **shaders** folder;
  - the **res** folder;
  - the program executable (**EDAF80_Assignment5**, from
    build/src/EADF80);
  - the **assimp** DLL (found in the same folder as above)
- Notes: for the **shaders** and **res** folders, you can ignore files you do not use as long as you keep the same folder hierarchy

Share your game!

- Then, zip it and share it!
- Here is an example below of a shared game:

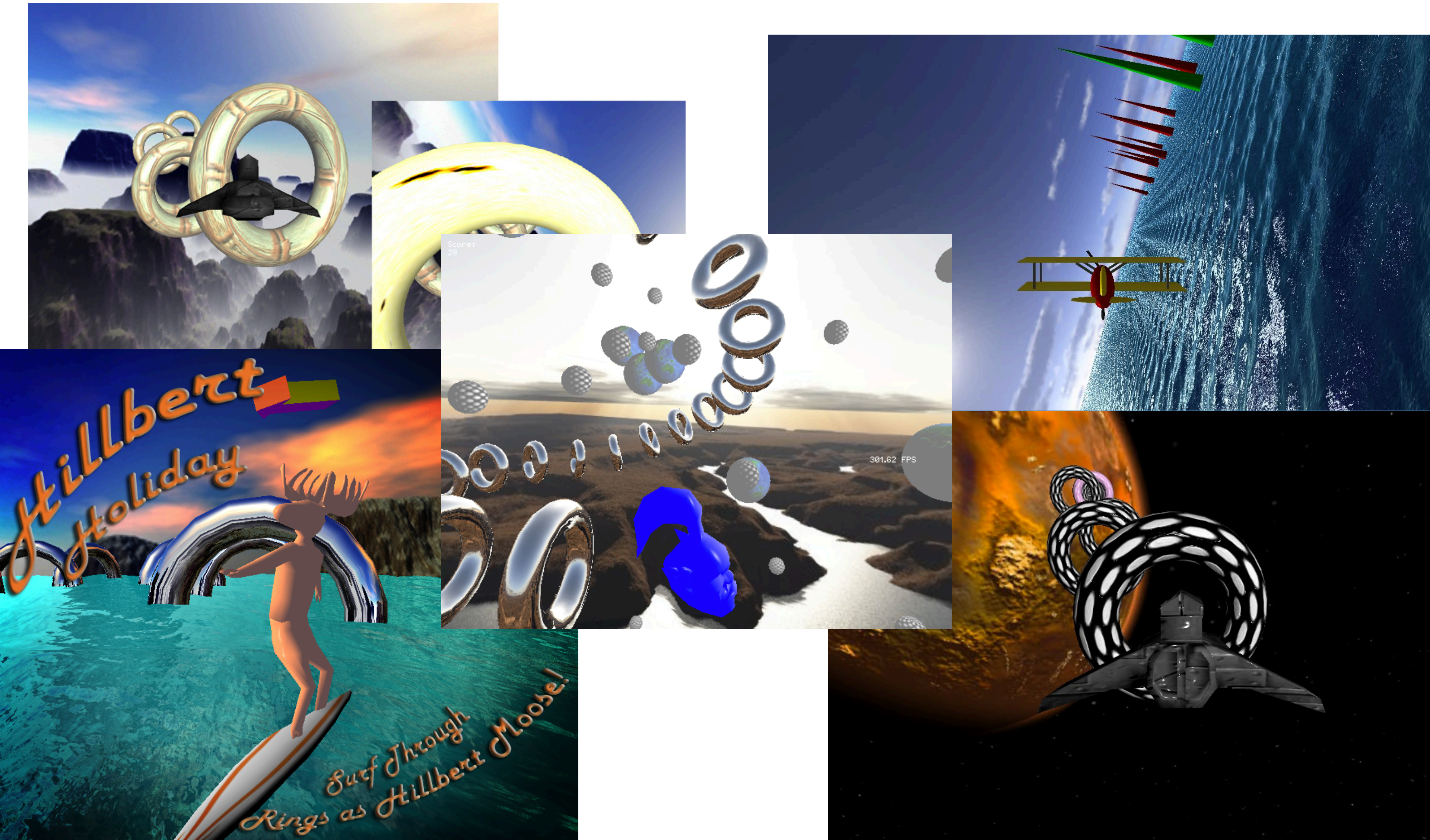


Name	Date modified	Type	Size
res	2016-10-14 09:40	File folder	
shaders	2016-10-14 09:50	File folder	
assimp-vc140-mt.dll	2016-10-02 16:38	Application extens	13 701 KB
Catmull Highway.exe	2016-10-14 13:32	Application	323 KB

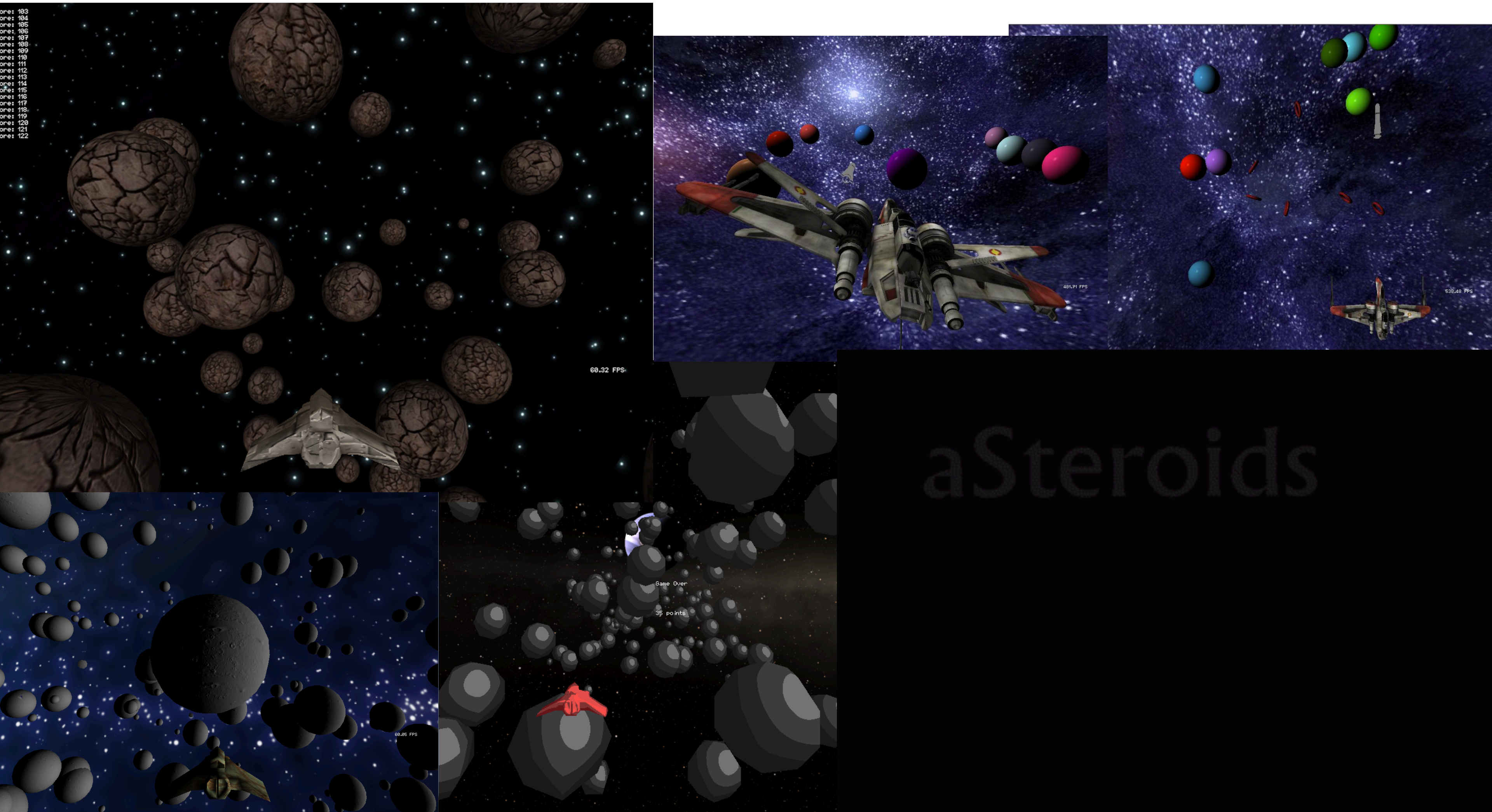
Today

- Final assignment: **make a game** ✓
  - Some ideas
- New stuff ✓
  - Collision detection
  - Physics (inertia)
- Miscellaneous helpers ✓
  - Add new files to the project
  - Load an external 3D model
  - Share your games with others

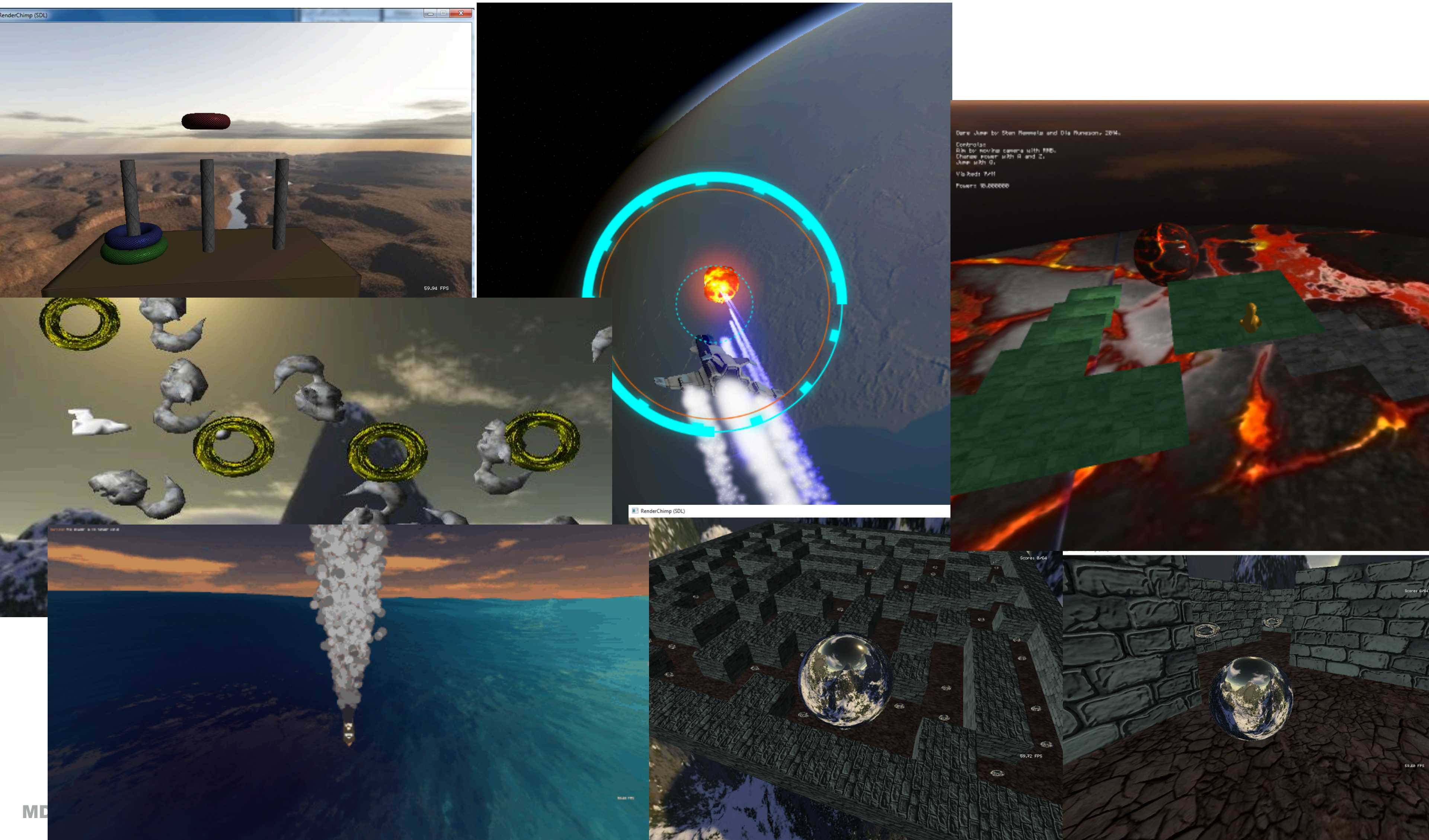
Torus ride examples



Asteroids examples

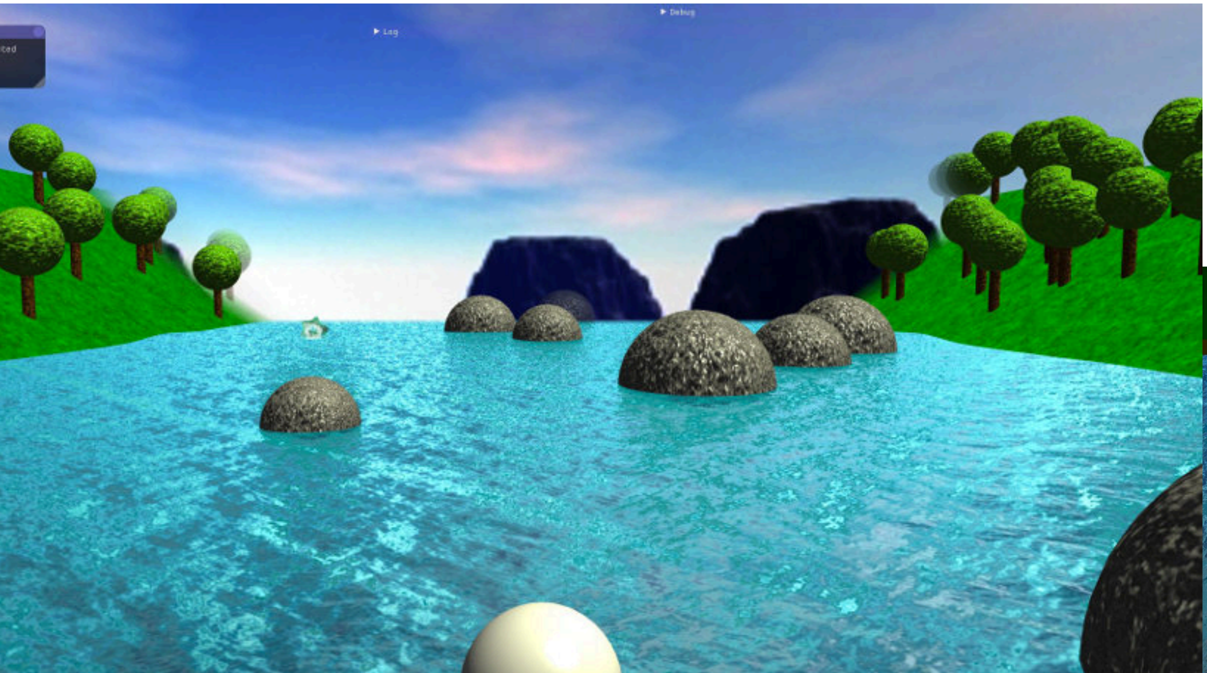


Other examples



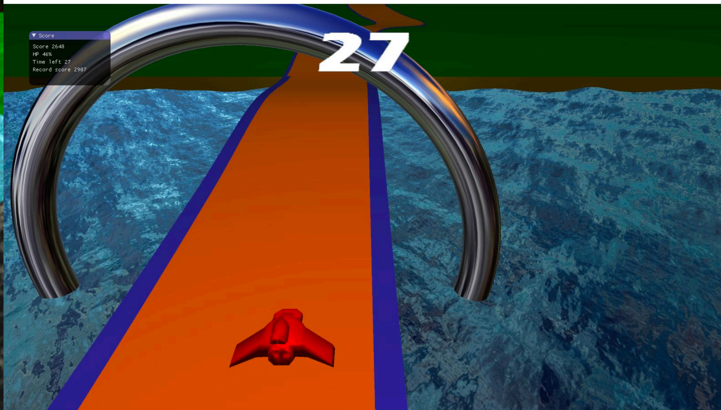
```
Dare Jump by Sten Remmelg and Ola Runesson, 2014.

Controls:
Aim by moving camera with RMB.
Change power with A and Z.
Jump with Q.

Visited: 7/11

Power: 10.000000
```

You collected 4 coins!



Catmull Highway

https://youtu.be/nG_bm0vBJ5Y



```
Score 2648
HP 46%
Time left 27
Record score 2907
```

General guidance

- Print scores, messages etc to console (printf)
  - even better, use ImGui
- Random numbers: `int rand(), #include <stdlib.h>`
- *Keep it simple*: start out with basic features, shaders etc
  - Add complexity progressively
  - Total time consumption equivalent to a normal lab
- Reuse your achievements from assignment 1-4

Summary

- Minimum requirements (Asteroids, Torus Ride)
  - Ship/camera manoeuvrability
  - Use of tessellated objects with shaders
  - Translational and rotational animation
  - Fixed object array (respawn if needed)
  - Game presentation on course forum
- Optional
  - Collision detection, Inertia, score count, ...
- Own idea
  - Consult us

Have fun & Good luck!