Programming Assignment – 2D/3D Computer Game

- groups of 2 students (in exceptional cases, 1 student)
- type/content: up to your own creativity!
- has to feature **4 techniques** from the lecture:

1a. Path interpolation
1b. Physically-based particle dynamics
1c. Mass-spring systems
1d. Rigid-body dynamics
2. Voronoi Fracture
3. Motion blur
4. Hierarchical transformations

At least two of the techniques 1a – 1d

Programming Assignment – Specs

- Webbased or Standalone Executable
- Has to run on a reference platform
(Win10, >8GB RAM, NVidia GTX >780, Visual Studio 2019, Firefox/Chrome)
- Web: HTML5, Canvas, PixiJS, ThreeJS …
- Standalone: C++ & OpenGL GLFW, SDL, SFML, …

- Frameworks/Libs may be used for asset import, controls, graphics
- No use of game engines, animation code (position and state updates of objects) has to be implemented yourself!
- Framerate-independent animations!
→ update of animation state decoupled from rendering update
- Code versioned via GitLab

Practical Assignment – 1. Submission

- Specifications as PDF (~3 pages), containing
  - Game title and name of your group
  - name and matr. number of the group members
  - description of the vision/concept of the game:
    - Content/Objects/Items, Aim of the Game, Controls
    - if given, reference to games that inspire your game
  - Specification of platform (web: which JS lib; standalone: which lang.)
  - at least one schematic mockup (hand-drawn is sufficient)
  - list of techniques you intend to implement and
  - description of how they add to the game (game mechanics, or just decoration)
- Specs per file upload (upload link t.b.a)
- Deadline: 12.03.

Practical Assignment – 2. Submission

- Running Pre-Release, featuring
  - basic input control (keyboard, mouse)
  - 3D-Games: running display-pipeline and camera control
  - Running animation loop (framerate-independent)
  - basic objects/primivites as game assets
  - at least one featured technique
  - short PDF commenting your submission (controls, techinque, code overview)
- Executable game files via zip upload (Link t.b.a.)
- Deadline: 30.04.

- **Note: By uploading the 2nd submission you will definitely receive a grade.**

Practical Assignment – 3. Submission

- Final Release featuring all specified techniques
- Executable game files via zip upload (link t.b.a.)
- Including a PDF containing:
  - User-Doc: Game-Manual (Controls, Gameplay ...), ~3 Pages
  - Tech-Doc: documenting techniques and where they are in the code

- Deadline: 11.06.

Exercise Interview & Oral exam

- ~60 min slots per group, registration via TUG-Online
- Interview on your implementation, live demo, questions about the code
- Oral Exam: 3-4 theoretical questions from the lecture

- Dates: 21.06. – 25.06.

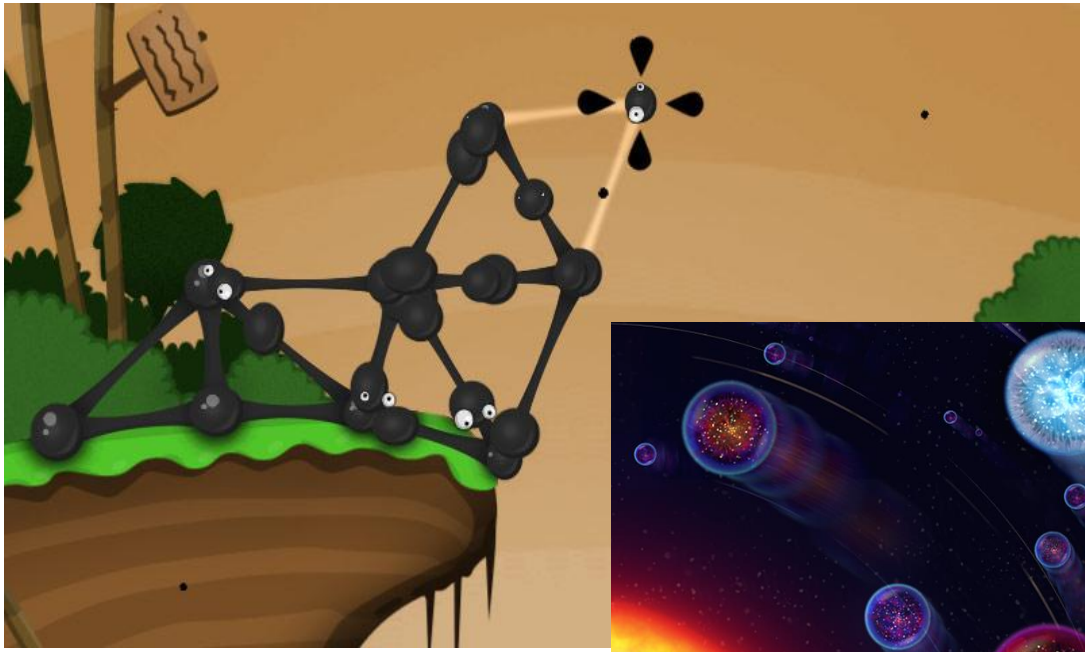
Game Event

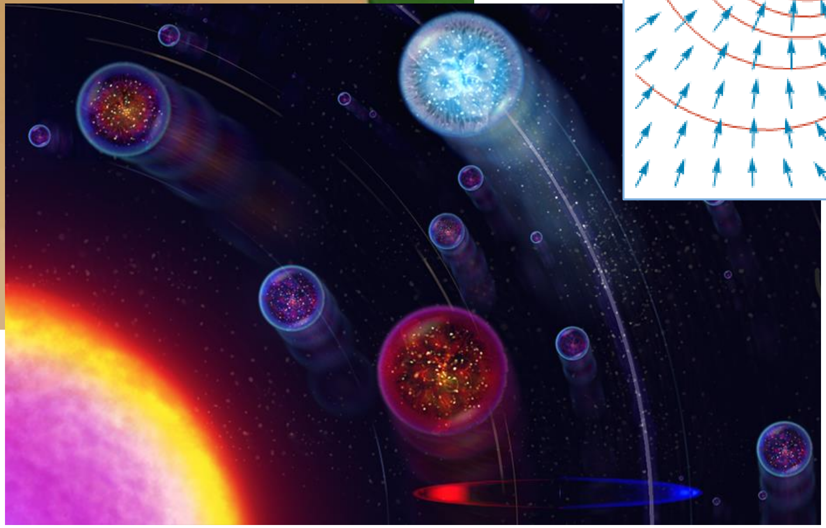
- ~10 min Virtual Live-Presentations of your games
- Preliminary Location: Discord

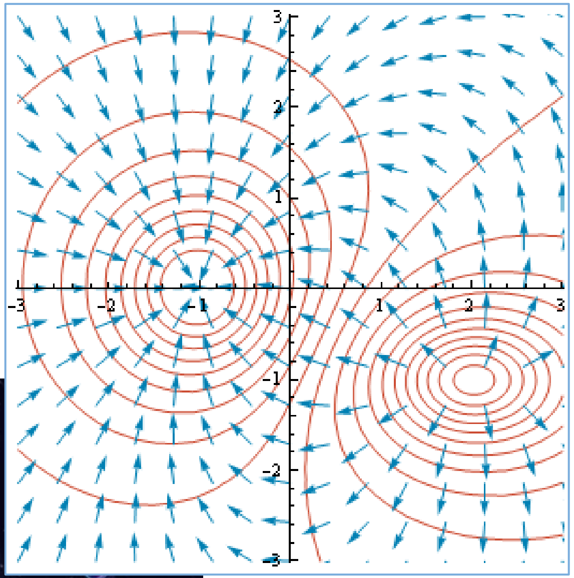
- Date: 28.06.

Passing the Course

- For a **positive grade**, you have to
  - **Pass the exercise interview (know and understand your code)**
  - **score 50% (20 pts) on the oral exam**
  - **score 50% (30 pts) on the assignment**
  - **score 50/100 pts in total**
- To improve on a negative grade, repetition of the oral exam is possible within 4 weeks after the end of the course
- Individual requests for a second exam per email to r.preiner#cgv.tugraz.at







Assignment Techniques

1a. Path Interpolation

- Your game contains the movement of one or more objects along a Catmull-Rom spline at controlled speeds (at least one non-constant motion, e.g., ease-in/ease-out).
- Spline evaluation is implemented by yourself
- Control points of the spline may be dynamic, manipulated by the user, …

Flight Control (Firemint)

1a. Path Interpolation (cont.)

- Required Visualizations (switchable)
  - *Spline curve*
  - *Control Points*
  - *Arc-Length Table Samples*
- Required adjustable controls
  - *Traversal Speed*
  - *Animation Update Rate*

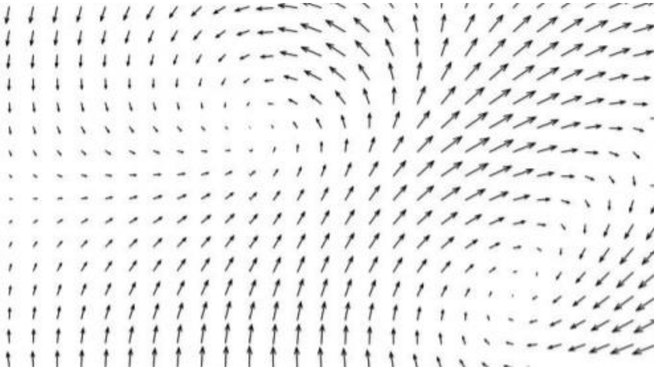
- *Practical learning targets:*
  - *Spline-Interpolation*
  - *Arc-Length Parametrization*
  - *Speed control*

  (→ Lecture 5. Interpolation)

Flight Control (Firemint)

1b. Physically-based Particle Dynamics

- Your game contains particle-like objects that move according to (Newtonian) physical forces.
- Implement an apropriate ODE solver that moves the particles according to the forces using **4th-order Runge-Kutta Integration**.
- Use at least three different non-global force sources:
  - Radial (gravity of multiple planets or mouse pointer)
  - Pre-defined force vector field (grid-based or analytic)



Osmos (Hemisphere Games)

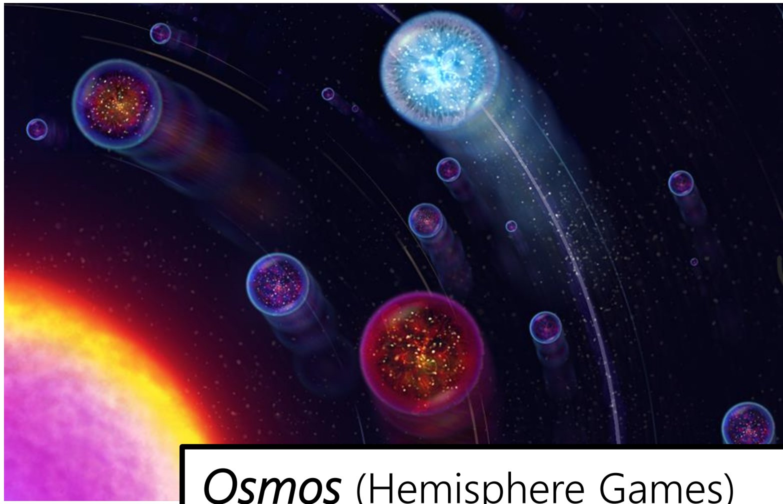
1b. Physically-based Particle Dynamics (2)

- Required Visualizations (switchable)
  - *Force field (vector grid)*
  - *Object trajectories of the last few seconds*

- Required adjustable controls
  - *Switch between RK-4 and simple Euler integration*
  - *Animation update rate / step size h*

- *Practical learning targets:*
  - *Force-based Particle Dynamics*
  - *RK-4 Integration*

  (→ Lecture 7. Physical-based Animation)



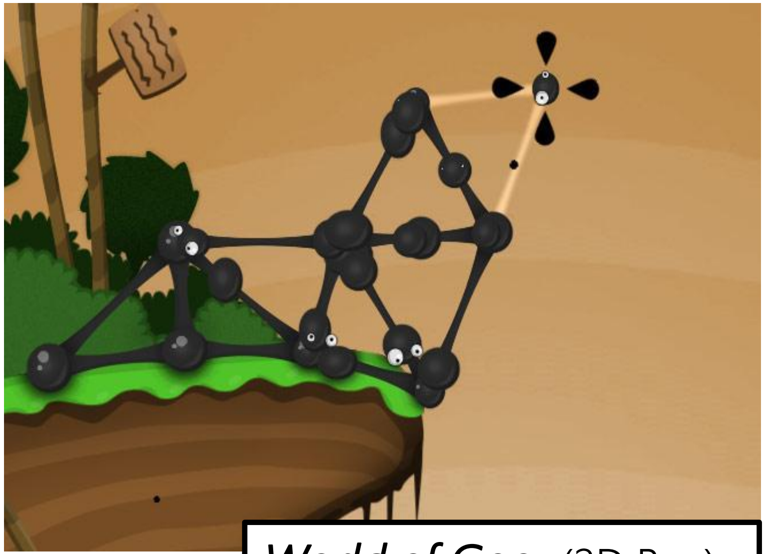
Osmos (Hemisphere Games)

1c. Mass-Spring Systems

- Your game contains a mass-spring system that interacts in some way with the user and behaves according to physical forces.

- You implement an ODE solver that moves the mass-spring object according to the given forces using **Velocity-Verlet Integration**.

- Your engine uses **Hookean forces** and at least **one external force** (e.g., gravity, wind, …)



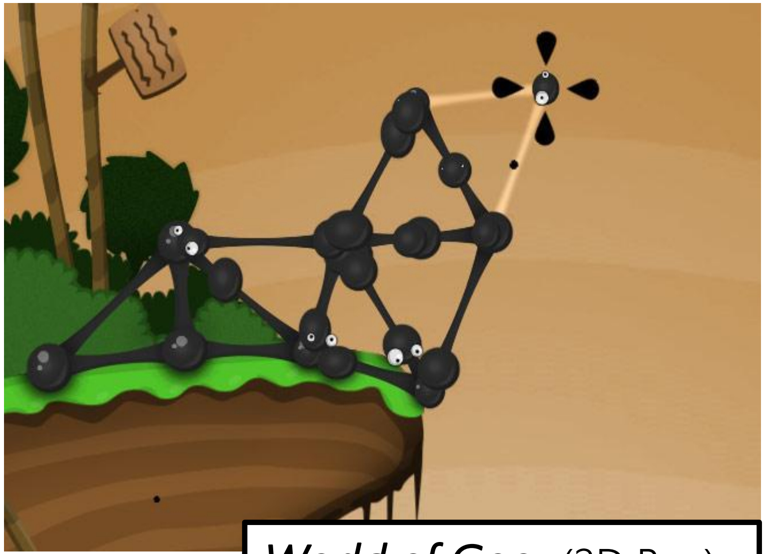
World of Goo (2D Boy)

1c. Mass-Spring Systems (cont.)

- Required Visualizations (switchable)
  - *Mass-Spring graph*
  - *Edges color-coded by spring strains*
  - *Current total force vector acting on each mass point*
- Required adjustable controls
  - *Spring Stiffness k*
  - *Animation update rate / step size h*

- *Practical learning targets:*
  - *Mass-Spring System Definition*
  - *Hookean Forces*
  - *Velocity-Verlet Integration*

  (→ Lecture 7. Physical-based Animation)



World of Goo (2D Boy)

1d. Rigid-Body Dynamics

- Your game detects collisions between finite objects that have linear and angular momentum.
- Objects react to **ellastic collisions** by changing their momentums based on physical forces (bouncing off, adjusting rotation, etc.)

- Required Visualizations (switchable)
  - *Momentum vectors (before and after collisions)*
  - *Collision points*
- *Practical learning targets:*
  - *Handling Elastic collisions*
  - *Linear and angular momentums*

  (→ Lecture 7. Physical-based Animation)

Angry Birds (Rovio)

2. Voronoi Fracture

- Your game contains 2D objects that are dynamically fractured into a number of small pieces using pixel-based *Voronoi Fracturing.*
- 2D objects (images) have a complex, **non-rectangular** silhouette
- Voronoi seeds of the fracture pattern are computed **dynamically** at runtime **based on given causalities** (bullet impact, punch location)
- Voronoi cells are compute based on a true pixel/grid-based distance field, which must be additionally **noised**
- Examples:
  - Shattered glass
  - Fragged enemies

Smash Hit (Mediocre)

2. Voronoi Fracture (cont.)

- Required Visualizations and Features (switchable)
  - *Voronoi Seed Points*
  - *Voronoi cell distance fields (color coding)*

- Required adjustable controls
  - *Additional Noise-Overlay on/off*

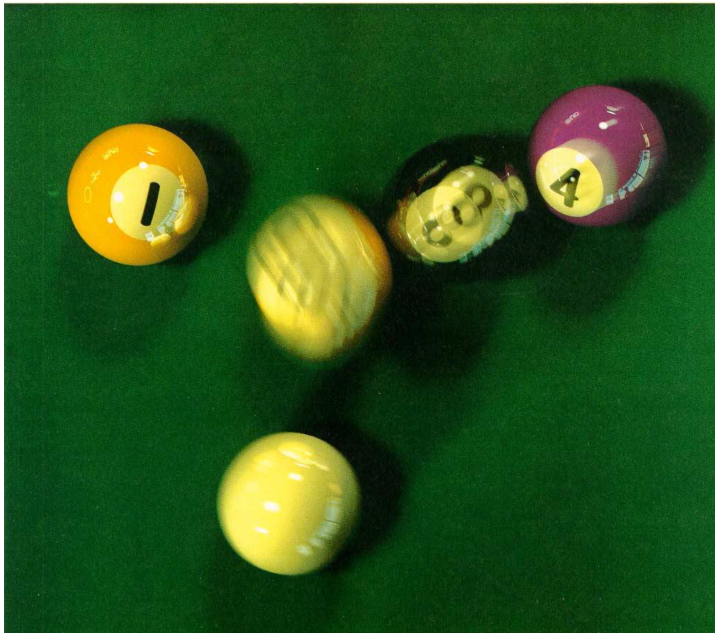
- *Practical learning targets:*
  - *Plausible Fracture Patterns*
  - *Implicit Shapes*
  - *Voronoi-Cells*
  - *CSG Operations*

  (→ Lecture 8. Voronoi Fracture)

Smash Hit (Mediocre)

3. Motion Blur

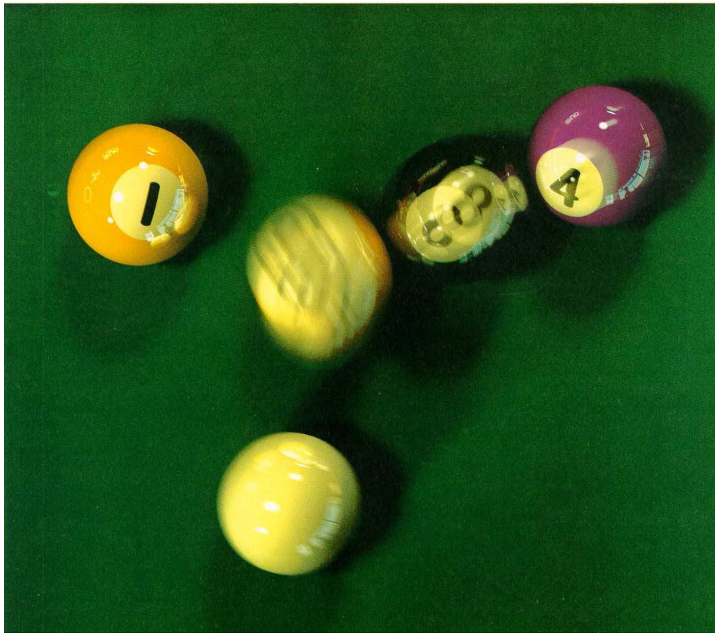
- Your game contains high-speed object movement that might not be properly resolved at slow animation update.
- You implement two motion blur strategies to counter temporal aliasing
  - stochastic motion blur
    OR
    post-process motion blur
  - supersampling
- allow to dynamically switch the mode at runtime in your game (differences should be visible)



3. Motion Blur (cont.)

- Required adjustable controls
  - *Switch between the two motion blur modes (differences should be visible)*
- *Practical learning targets:*
  - *Temporal anti-aliasing*
  - *Motion blurred animations*

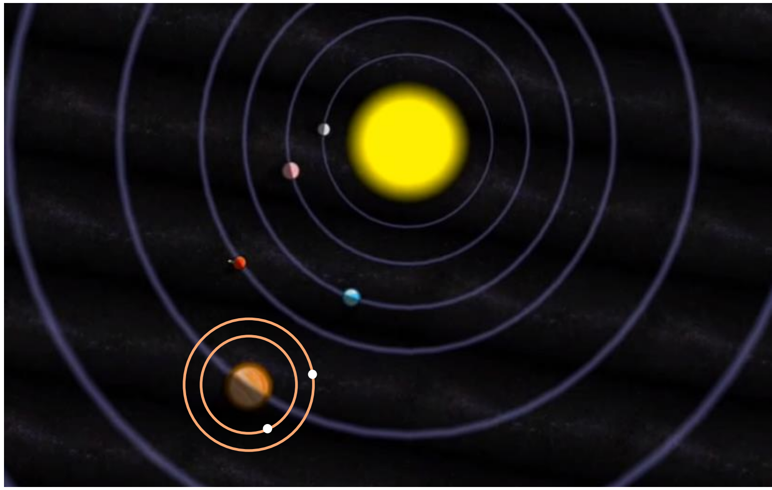
  (→ Lecture 3. Motion Blur)



4. Hierarchical Transformations

- Game objects move relative to other parent objects
  → hierarchy of reference systems
- Your game manages a hierarchy of transformations **(translations and rotations, at least 3 levels after root)** and propagates parent transformations down to its children
- Each level contains multiple objects
- Objects and their transformations can be controlable by the player.
- *Practical learning targets:*
  - *Hierarchical Scene Graphs*
  - *Transformation Representations*

  (→ Lecture 1 and 2)

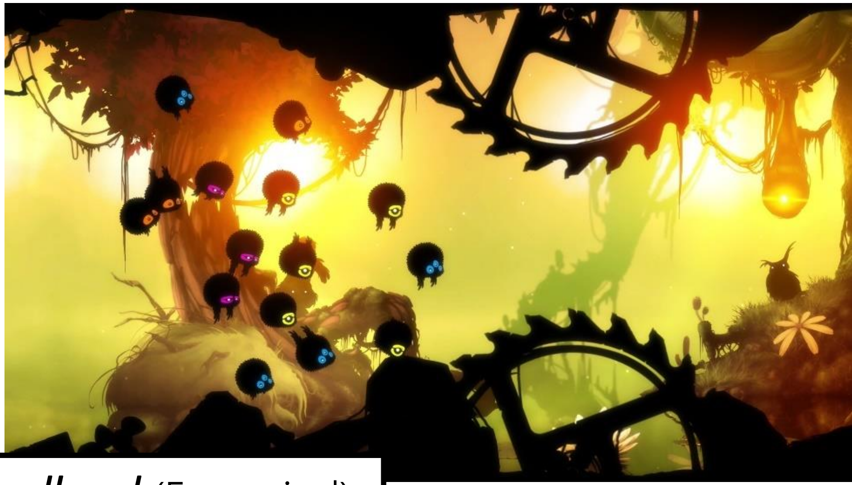


General Requirements

- Fully implemented gameplay (game-over, scores, etc.) not required for full assignment points, will however bring bonus points.
- Frame rate and simulation/animation update rate should be dynamically adjustable.
- Game pause (freeze animation and screen) and continue should be possible at any time.
- When using assets (images, sprites, sounds) from external sources, all sources have to be documented!
- **Only FREE assets are allowed!**

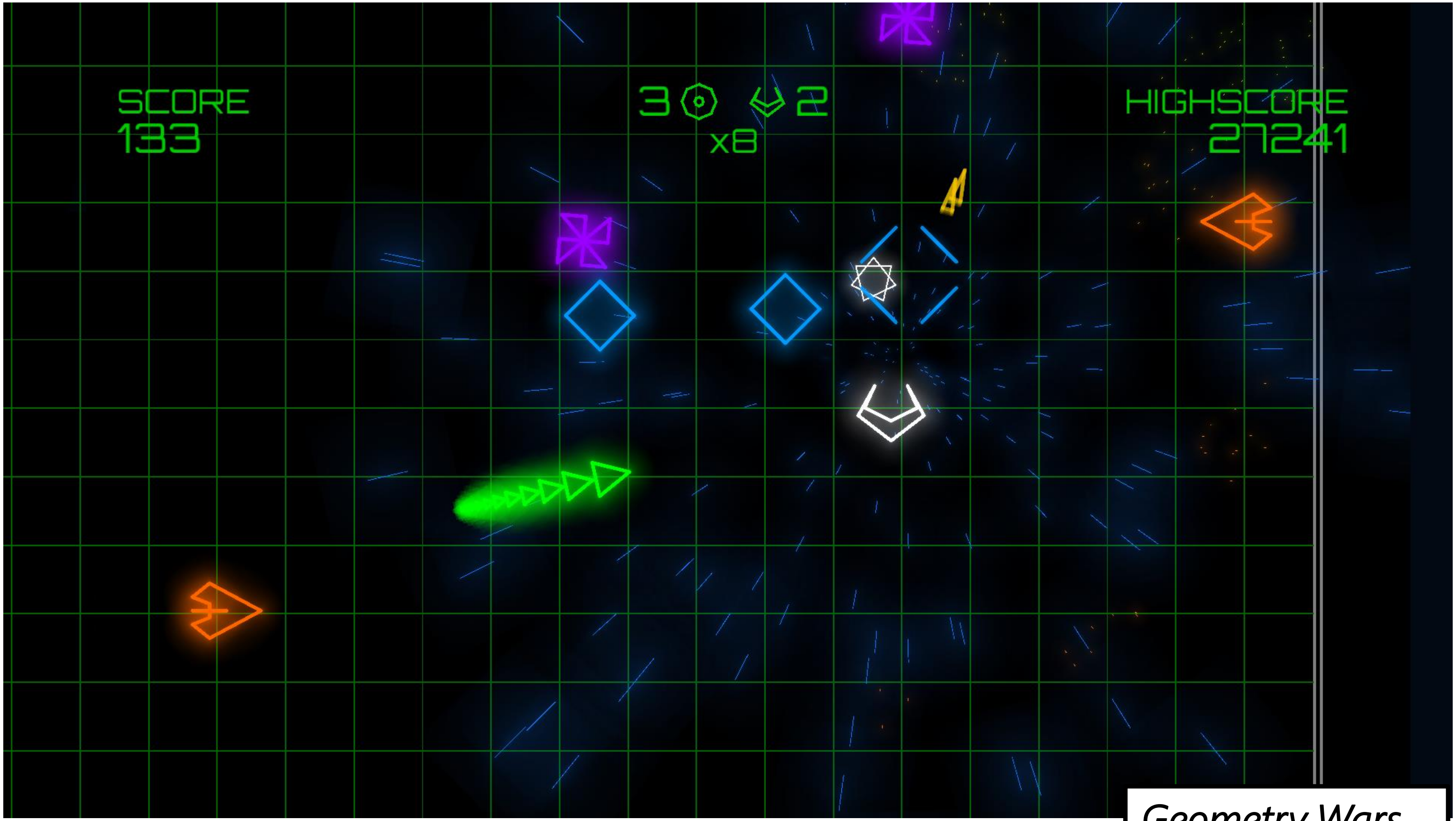
Hints

- Visual are Secondary!
  - Focus on game mechanics and S&A techniques first, then graphics
  - Start with primitive mockup shapes, replace later
  - Or use simplistic visual style altogether
- Sounds are appreciated, but not mandatory
- Webbased Apps are probably easier to start with
- Standalone apps allow more flexibility



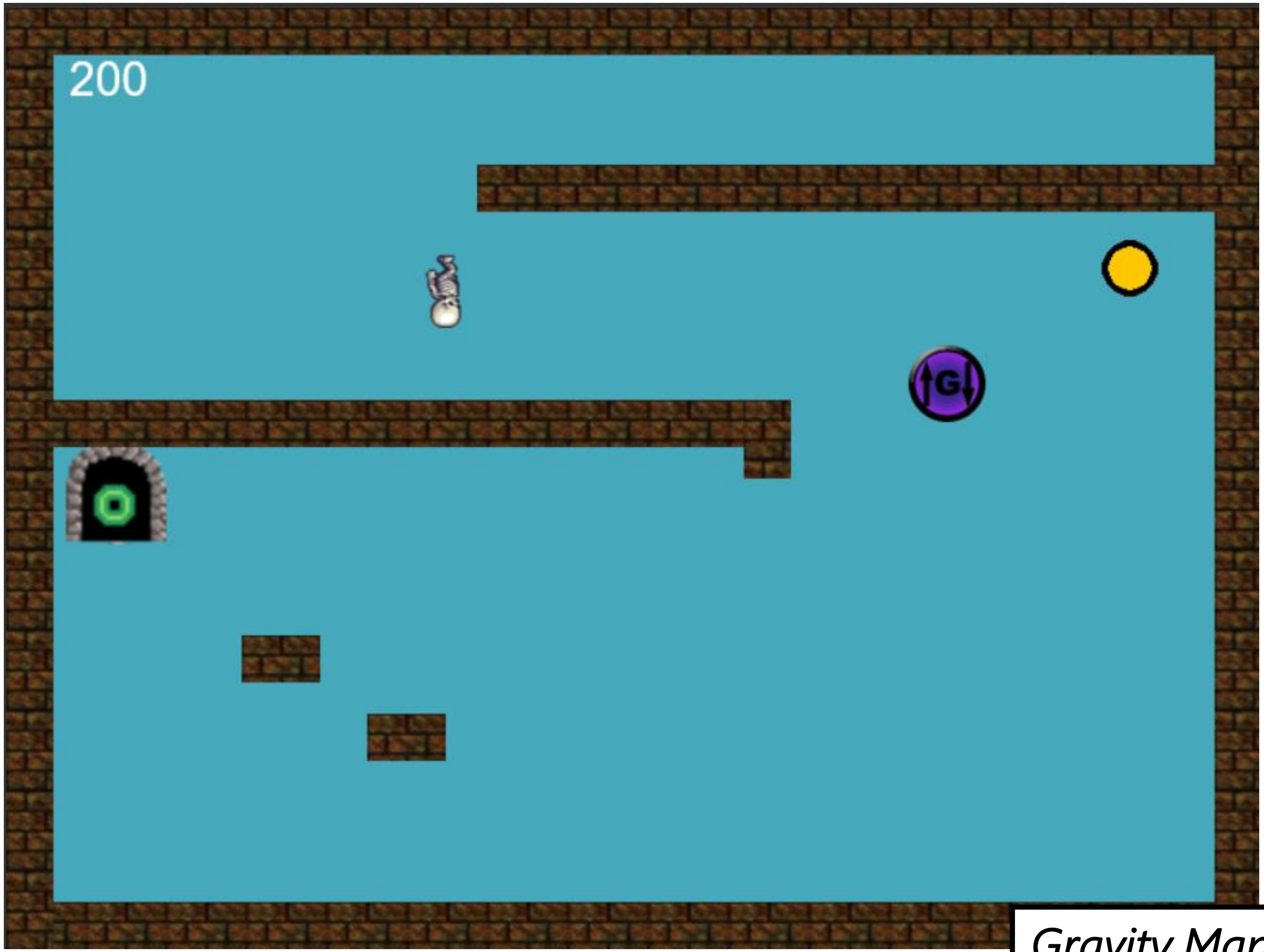
Badland (Frogmind)

Impressions from previous years



Geometry Wars

Impressions from previous years



Gravity Man

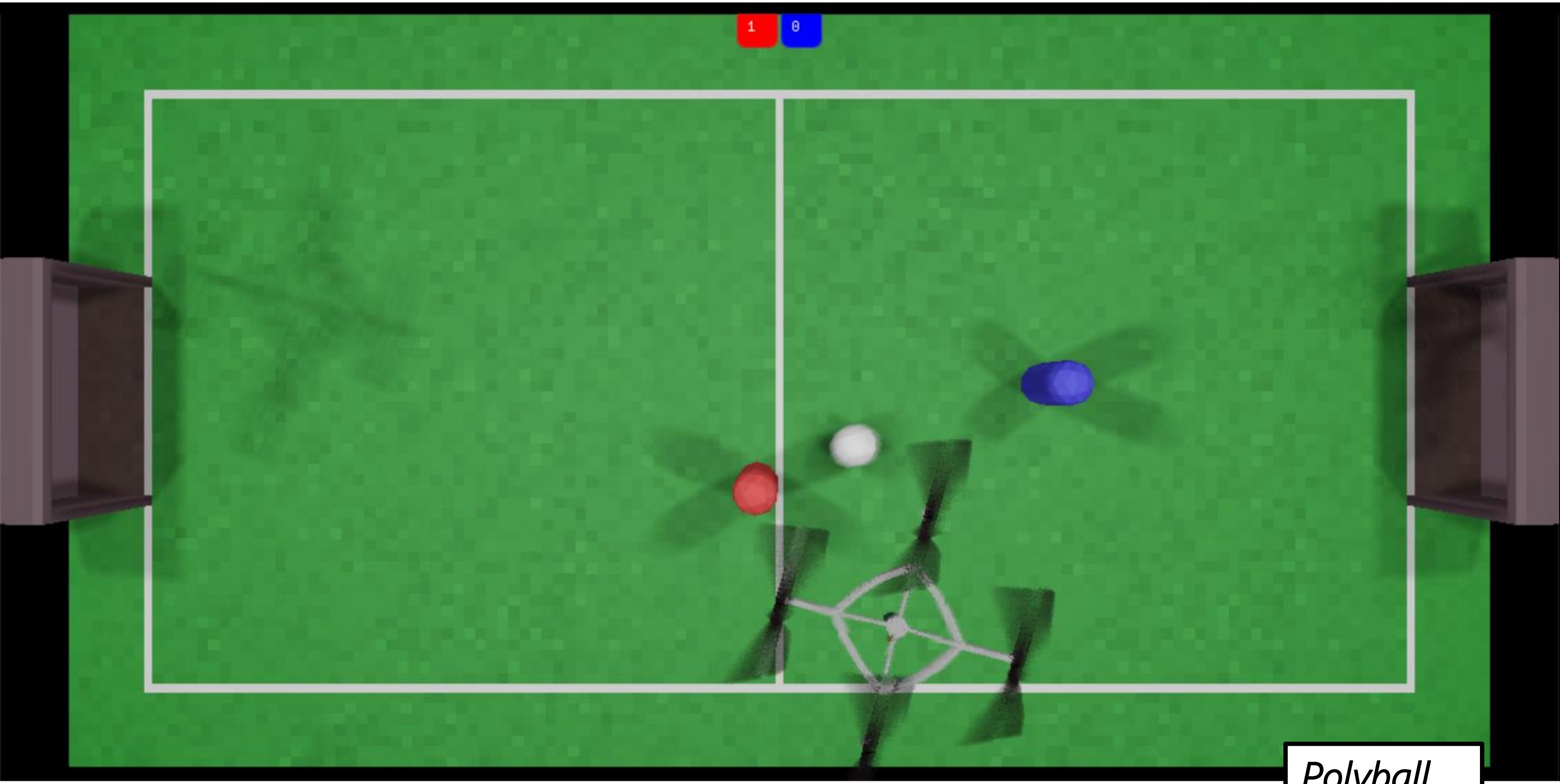
Impressions from previous years

Planet Pool

Impressions from previous years

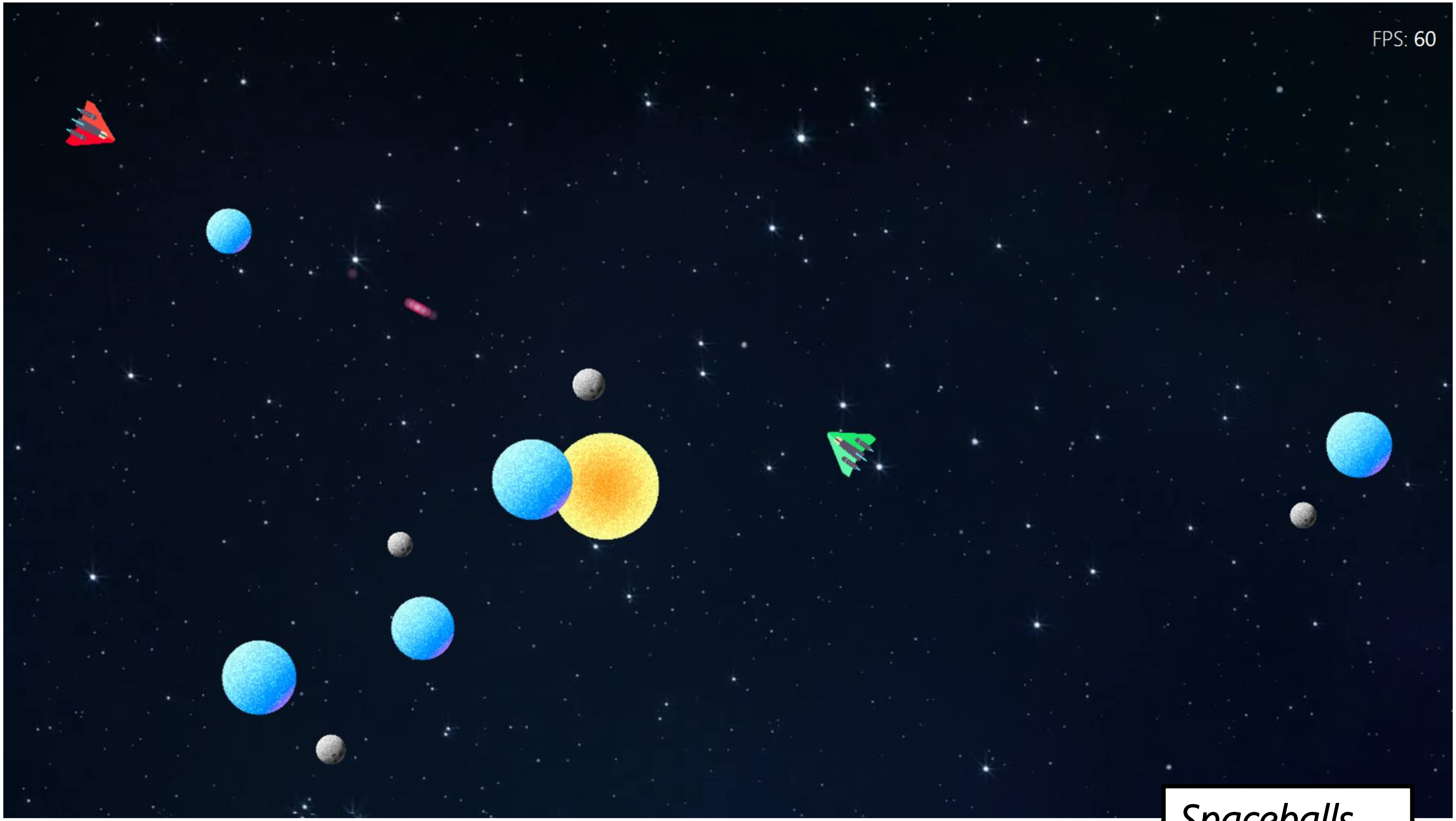
Under Construction

Impressions from previous years



Polyball

Impressions from previous years



Spaceballs

Impressions from previous years

Intelligent Space Missiles

Links

WebGL

- HTML5/Canvas:
  - www.w3schools.com/html/html5_canvas.asp
  - www.html5rocks.com/en/tutorials/canvas/notearsgame
- PixiJS www.pixijs.com
- ThreeJS www.threejs.org

C++/OpenGL

- GLFW www.glfw.org
- SFML www.sfml-dev.org/tutorials/2.4
- SDL www.libsdl.org
- GLM glm.g-truc.net
- Eigen eigen.tuxfamily.org