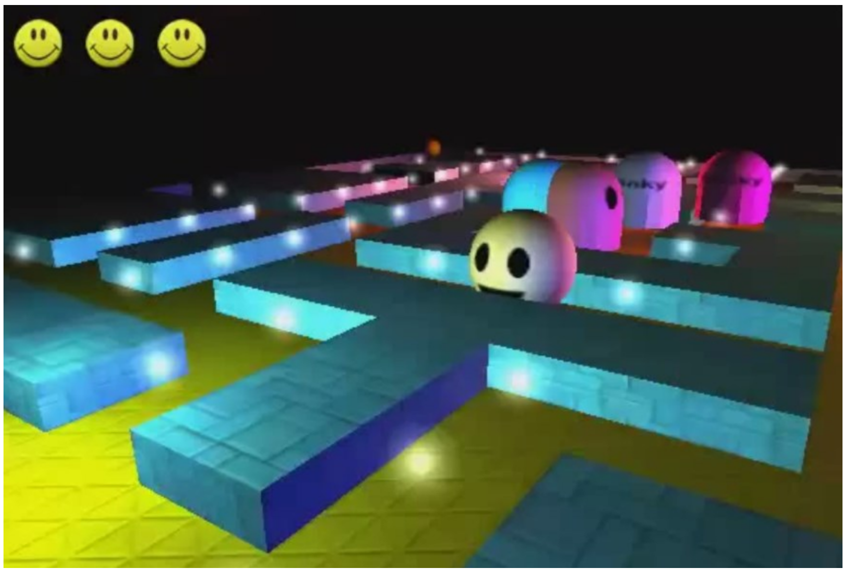
ΠΛΗ 412 Graphics Project, 2015, PacMaze3D! Instructions

The task is to create a 3D clone of the popular game Pacman using WebGL (WebGL - OpenGL ES 2.0 for the Web http://www.khronos.org/webgl/) and JavaScript only. The task is intended to demonstrate simple and fast cross-platform interactive application development using state-of-the-art technologies.



The task consists of four sections:

1. Design of the game mechanics and level representation (30%)
2. Design and rendering of geometry and basic functionality (30%)
3. Extended functionality (30%)
4. Personal touch (10%)

The percentages indicate an approximate division of marks for each section of the practical work. The division of marks should also guide the development effort.

The completed practical section will require each of the following files to be included in the submission (please give your name to the .zip file):

- The modified sandbox with all .js files (if you added any)
- Texture files
- Technical Report

Resources

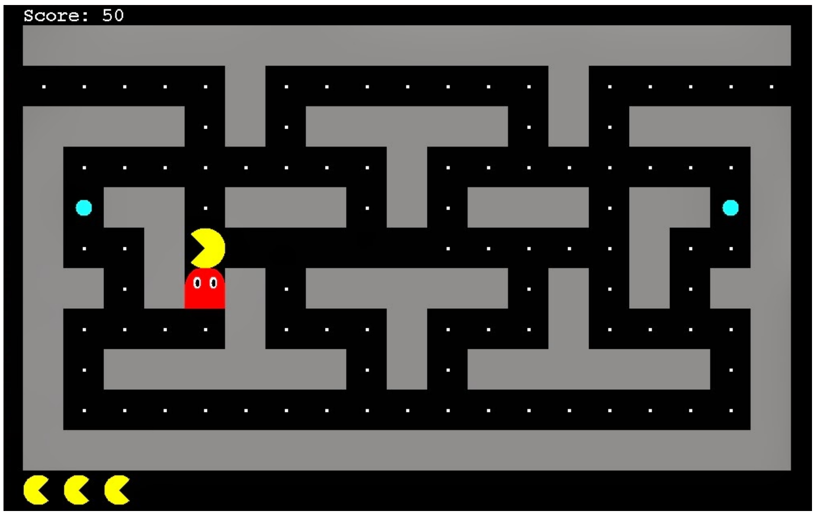
- Pacmaze demo: https://www.dropbox.com/s/61dpt6vdhgmilb1/Pacmaze%204%20screencast.mp4?dl=0

- Learning WebGL Tutorial set: http://learningwebgl.com/blog/?page_id=1217
- JavaScript Tutorial: http://www.tizag.com/javascriptT/index.php
- HTML Tutorial: http://www.w3schools.com/html/html_intro.asp
- The provided sandbox + All code comments (TUC Courses Platform)
- + every document in TUC Courses platform

1. Design of the game mechanics and level representation

The player controls Pac-Man through a maze, eating pac-dots (also called pellets). When all pac-dots are eaten, Pac-Man is taken to the next stage (you will design at least 2 stages!). Four enemies, Blinky, Pinky, Inky and Clyde roam the maze, trying to catch Pac-Man. If an enemy touches Pac-Man, a life is lost and the Pac-Man itself withers and dies. When all lives have been lost, the game ends. Near the corners of the maze are four larger, flashing dots known as power pellets that provide Pac-Man with the temporary ability to eat the enemies. The enemies turn deep blue, reverse direction and usually move more slowly. When an enemy is eaten it is later regenerated in its normal color. Pacman is controlled with the up/down and left/right arrow keys. You will use the WebGL sandbox that we provide you to implement the game. You are free to implement any extra functionality if you want.

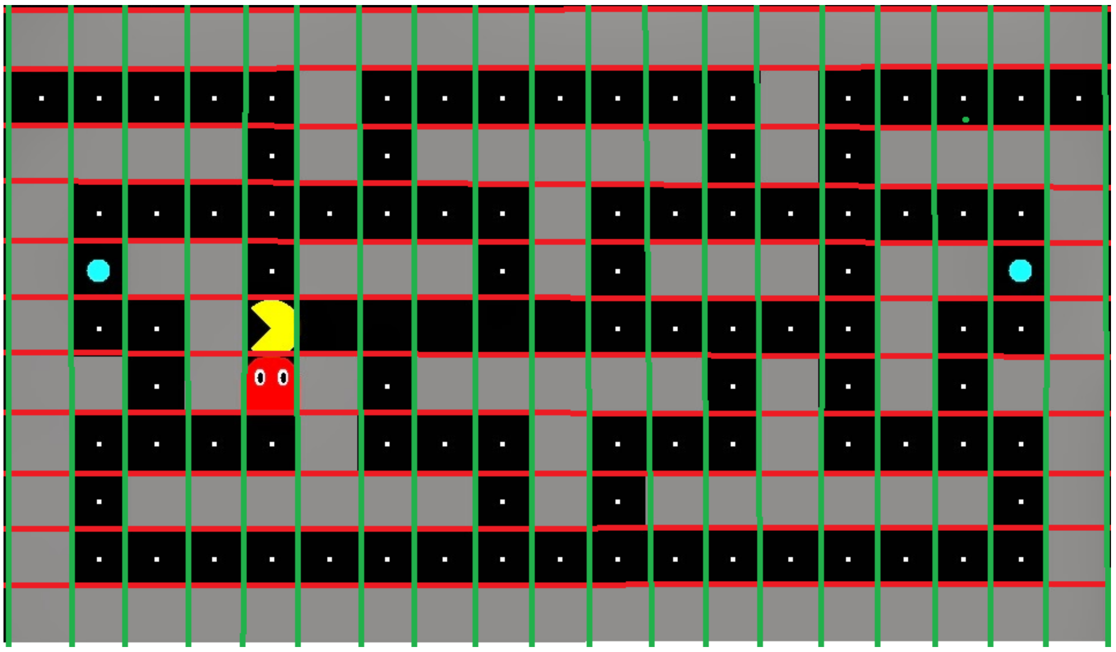
The sandbox has already sample code for transformations and a spinning cube that can be controlled with the arrow keys. It also includes all the debugging code required to find and correct mistakes. You have to create a new drawing function either by loading a model or by drawing with WebGL primitives. The maze, pellets and power pellets, will be positioned on the level according to level maps hardcoded in your code.



For example for this **2D version** of Pacman, the following 19x11 grid representation describes the level above:

```
x x x x x x x x x x x x x x x x x x x
p p p p p x p p p p p p p x p p p p p
x x x x p x p x x x x x p x p x x x x
x p p p p p p p p x p p p p p p p p x
x U x x p x x x p x p x x x p x x U x
x p p x p p p p p p p p p p p x p p x
x x p x p x p x x x x x p x p x p x x
x p p p p x p p p x p p p x p p p p x
x p x x x x x x p x p x x x x x x p x
x p p p p p p p p p p p p p p p p p x
x x x x x x x x x x x x x x x x x x x
```

Hint (x: maze blocks, p: pellets, U: (power pellets)



In your implementation you will employ **a 2D representation of the scene that you will <u>render in 3D similar to the figure on the first page.</u>** You may have as many levels as you wish; this project requires from you to have **<u>at least two</u>**. The mechanics of the game must check for every move, according to the current position of Pacman, enemies, pellets and power pellets what happens next according to the gameplay.

2. Design and rendering of geometry and basic functionality

Having a level representation and game mechanics in JavaScript you need to render Pacman, enemies, pellets and power pellets. You are free to use the colors you like and design the levels as you wish. Note that carefully designed levels make the game more enjoyable. **You will implement two cameras.** A panoramic camera, showing the entire level and a gameplay camera that follows Pacman.

3. Extended functionality

Extended functionality includes textures, animated cameras and better lighting (10% each).

Textures

You can add textures on Pacman, maze or floor. Example code on texturing can be found here: http://learningwebgl.com/blog/?p=507. Remember to include the textures on the deliverable. Check important notes section for texturing.

Animated camera

You can make the movement of the camera more sophisticated by animating it when the game state changes, e.g. a power pellet is consumed it moves from one position to another see https://www.dropbox.com/s/61dpt6vdhgmilb1/Pacmaze%204%20screencast.mp4?dl=0.

Lighting

You can use the more sophisticated Phong shader to light up your scene instead of just ambient coloring provided by the sandbox. A tutorial can be found here: http://learningwebgl.com/blog/?p=684.

4. Personal touch

Feel free to add any functionality you want to make a more personalized game. Implement **at least two** features either from the list below (of any difficulty) and/or something you like (color coding for difficulty level).

- ✓ Audio
- ✓ Themes
- ✓ Menu
- ✓ Extra Camera angles
- ✓ Random level generation
- ✓ Smarter enemies based on: http://en.wikipedia.org/wiki/Pac-Man#Enemies
- ✓ Model loading (http://learningwebgl.com/blog/?p=1658)
- ✓ Shadows (http://o3d.googlecode.com/svn/trunk/samples_webgl/o3d-webgl-samples/shadow-map.html)
- ✓ A*

Alter the game play as you wish (as long as you implement the basic functionality that we ask you to), use HTML to provide the player with drop down lists to select a level, difficulty, colors. The sky is the limit… ☺

Deliverables

- The modified sandbox with all .js files (if you added any)
- Texture files
- Technical Report

(please give your name to the .zip file)

Important Notes

You are advised to use Google Chrome for testing and Notepad++ for code writing. Remember before every refresh (F5) of the web page to save (Ctrl-S) your code in Notepad++!

DO NOT use any scene graph management framework, e.g. ThreeJS. Textures must be power-of-two, eg. 64x64 pixels, 128x128 etc

Remember to add comments in your code where required.

- JavaScript and ESSL comments are added with // and /* */ notation
- HTML comments are added with the <!-- --> notation

Useful Shortcuts

- Ctrl + U shows code in Chrome
- F12 opens Console for debugging
- If you want more advanced debugging use WeGL Inspector: http://benvanik.github.com/WebGL-Inspector/

For remote texture support choose one of the following.

- (suggested!) You can download the latest version of Python. Install it and choose add to path. Then you open a console window in your project directory and execute:
  ```
  python -m http.server
  ```
  Then in your browser:
  ```
  localhost:8000/<yourpage.htm>
  ```
- You can upload your project in your personal TUC space with FileZilla and execute from there. The textures will be "remote" and chrome will not block them. Instructions: http://www.tuc.gr/208.html
- Add `--allow-file-access-from-files` to Chrome shortcut to allow for local file access (does not always work!)

Good Luck!!!