Game Dev: Quake Console
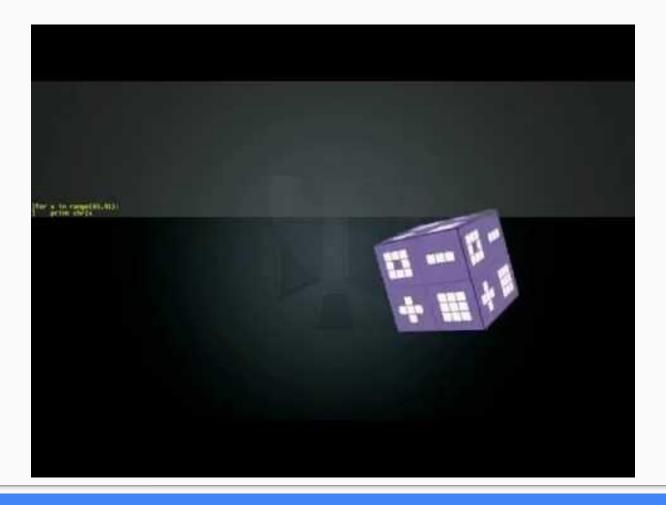
Ricard Pillosu - UPC

Check solution.exe for a WIP console implementation

```
name: meta2 firstgid: 25
tile width: 64 tile height: 64
spacing: 0 margin: 0
Layer ----
name: Graphics
tile width: 25 tile height: 25
Layer ----
name: Colisions
tile width: 25 tile height: 25
render start
j1App::Start took 98.021900 ms
UPDATE PHASE ===============================
Unable to render text surface! SDL_ttf Error: Text has zero width

Loading new Game State from save/save_game.xml...
...finished loading
Saving Game State to save_game.xml...
... finished saving
map, quit, list
```

quit

Quake Console

Crysis Console

Custom Engine Console

General Implementation

- Create a new Module for the Console

- It should use the UI Module to create needed Labels and Input Text

- Listen to input so on tilde/ESC the console should open / close

  - Tilde should be SDL_SCANCODE_GRAVE

- It should accept scroll with center mouse

- Re-route all LOG entries to the output

- You can have it define its properties (size, colors, font) in config.xml

Implementing Commands

- Create a small class that represents a command

  - Which properties would you use for it ? (name, amount of arguments, … ?)

- The console should process input the same way Zork parsed commands

- Now each module should be able to generate its own commands:

  - Module Render: turn on/off sync, fullscreen, etc..

  - App: quit, max fps, pause, load level, etc…

- Have a function to generate a new command

```
App->console->AddFunc(const char* command, j1Module* callback, uint min_arg, uint max_args);
```

Extra Features

- Scroll with pgUp/pgDown

- Allow mixing uppercase and lowercase as commands

- Pressing UP arrow should write the previous command in history

- Pressing DOWN arrow should write the next command in history

- Pressing TAB should try to autocomplete
  - Only if there is 1 hit while you search for commands, if not make a beep with **printf("\a");**

- Allow argument packing with quotes

```
]log "hello world" ERROR
```

Homework

1. Create a console that appears on a keystroke

2. It should contain a scrollable text with all available LOG

3. Have a subsystem to create and parse commands

4. Create the following commands (with Module responsible):

   a. **log** (App): will log all arguments as one string

   b. **quit** (App): exit successfully the application

   c. **help** (Console): list all console commands

   d. **vsync** (Renderer): turn on/off vsync (e. g. vsync 0/1)