Project II - Personal Research

Goals

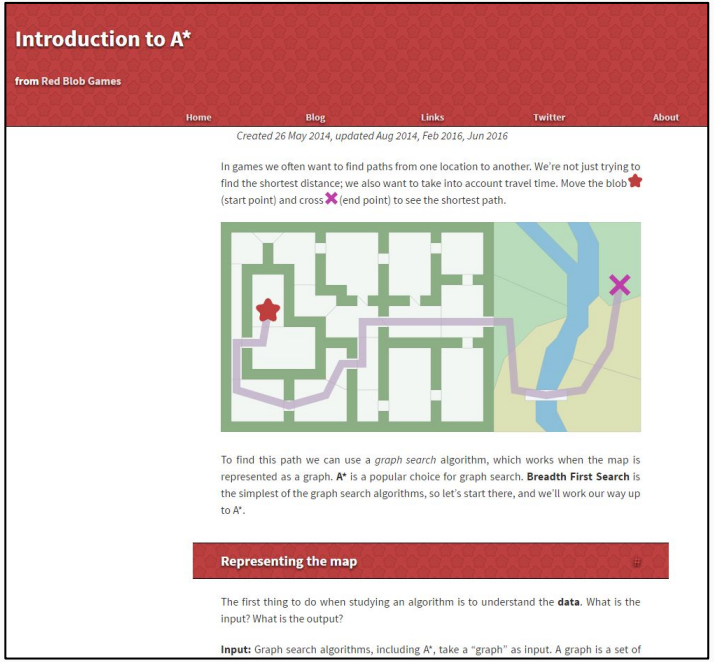
- Learn how to find, solve and assimilate any problem by yourself
- Practice communication performing teacher duties for a day
- Create simple yet generic code that can be used anywhere else
- Become expert in a small area and attend other students request
- Help the games that are being developed with new code modules
- Have a new element to add to your CV

Website

- You need to create a website under github.com with:
  - Intro to the problem
  - Different approaches by different games *(use animated gifs when possible)*
  - Description in detail for the selected approach
  - Explanation of any other improvements on the system
  - TODOs and Solution inside the repository as VS projects
  - Optional Homework for practicing
  - Links to more documentation

Website

- The website should be a tutorial that explains the problem and solution with as much visuals as you need.
- Focus on simplicity, your audience is not only the class but all aspiring game developers.
- Examples here.



Website

The header of the website should contain the following signature:

“I am <link to your linkedIn>(NAME LASTNAME), student of the <https://www.citm.upc.edu/ing/estudis/graus-videojocs/>(Bachelor’s Degree in Video Games by UPC at CITM). This content is generated for the second year’s subject Project 2, under supervision of lecturer <https://www.linkedin.com/in/raysan/>(Ramon Santamaria).”

Pure Theory presentations

- Initial presentations can involve only theoretical concepts.
- Material provided should be **useful** for all teams in their development.
- All student should learn **something new**, avoid explaining obvious concepts that students used to play games already know.
- **Never** copy content without a citation

Code

1. Solution that shows all the TODOs finished. **Only code with STL!**
2. Handout with 3-8 small TODOs from easy to difficult
    a. All TODOs should be easy to find in the code base with comments to support them

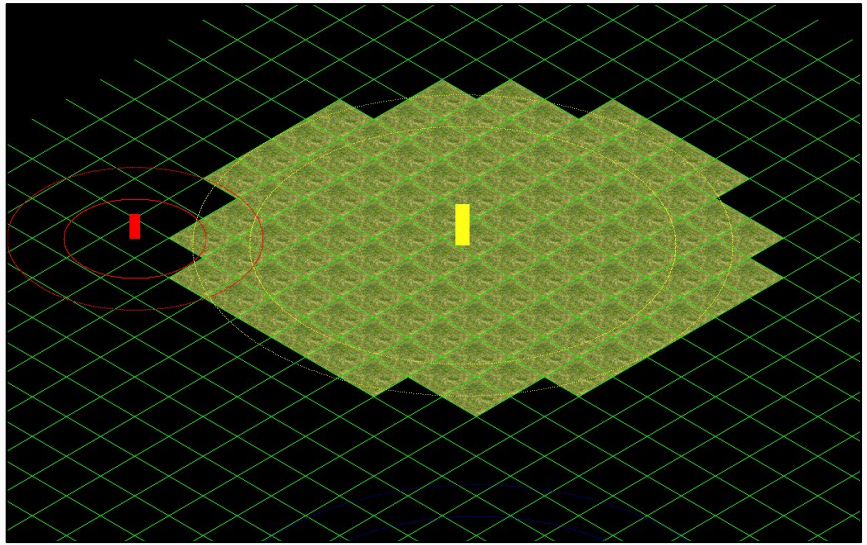
Base code should be as small as possible, striping any other not-used code if possible (same for assets). Code should express the core solution in a **simple** and **understandable** manner. Include as many debug visualizations as possible.

Repository

- Everything should be in a repository:
  - /docs - folder for all website material
  - /exercises - code with the TODOs in comments
  - /full_code - solution (keep the TODO comments)
- It should include a build published in the **Release** to download as zip
  - Only the executable with everything solved and enough debug visualization to understand the code and how it works (explained in the README)
- Measure your module performance in **ms** and discuss potential optimizations, with links to point out how to do those techniques

Generic Solutions with generic code

GOOD



BAD

Timing

- Teacher will provide a full calendar with all research areas
- Research might be presented to the teacher the previous class before presentation for approval / improvements
- Practice the presentation time and approximate TODOs exercises to last no more than 30/45 minutes
- The material as well as the presentation skills are evaluated in the grade
- Link must be uploaded before starting the class to the proper folder

Research areas

- Teacher can help and guide you *after* you start searching.
- Most common mistakes:
  - Concepts are too obvious.
  - TODOs are too complex / long to finish.
  - Initial explanation of the problem and solution is done too quickly without clarifying well each concept.

RESEARCH AREAS

GAME DESIGN

DOCUMENTATION

CODING

BUILDING

Area: Game Pillars

- Games production always start with a set of **3-8 core pillars**
- Those act as filters to discard new ideas during development
- Which game pillars were defined in professional games?
- What would be a bad game pillar? How is this useful for our teams?

Area: Quest Design

- How are quests designed in professional teams?
- Example of Good/Bad Quest design
- Quest as a learning / tutorial experience
- Dangers of grinding
- **Avoid the obvious concept all students already know.**

Area: Boss Design

- How are bosses designed in professional teams?
- How they affect the overall level design?
- What they mean for the player?
- Bosses as a test for players skill
- Boss themes

Area: Tutorial Design

- Examples of good/bad tutorials
- Learning curve analysis
- **First 3 minutes of game**
- Tutorial formats used

Area: RTS Balancing

- Base for paper/scissor/rock
- Expectation of strategic gamers
- How much strategy vs tactics/skill
- **Approach to balancing maths**

Area: Production Plan

- Production plan is a document that needs to be delivered
- What does it contain?
- How and why is it useful?
- How and when it should be updated?

Area: Tech Design Document (TDD)

- It defines all **technology related items** that should be considered
- Code style, platforms supported, code organization overview (UML)
- Functionality and limitations expected to communicate to the team
- Branch workflow, code reviews, performance budgets

Area: Art Bible

- Art Bibles are created by Art Directors before entering production
- Contains guides for both visuals and production limits (texture size, etc.)
- How is this useful for our teams? Examples of professional art bibles
- Items commonly defined in Art Bibles: palettes? character references?

Area: Audio Bible

- Audio Bible is a document that should be delivered
- It mirrors the Art Bible but for audio
- How it affects the creation of audio?
- How it defines the production of music?

Area: QA Workflow

- QA uses workflow diagrams to define its pipeline
- Those change from development phase to phase
- How balancing secure vs. agile workflow
- What about Quality Test? Stabilization phases?

Area: Available Licenses

- Dive into the legal aspect of software and game development
- What is copyright? What is trademark?
- Free software licenses teams could use and their differences
- Licensing IPs

Area: Quest Manager

- How to organize internally player objectives
- Prerequisites: other quests finished? some stat (level)?
- Progress track: event system that captures, e. g. inventory changes
- Data driven quest creation with XML

Area: Dialog System

- Intro to different dialog systems in games, from simple to complex
- How to remember previous choices? How to substitute player name?
- Solid code that can quickly choose between options
- Ways to define the data (XML)

Area: Particle System

- Intro to particle systems, focus on 2D
- Explanation on the common particle properties
- Solutions available out there (check particle2dx.com)
- At least implement a good looking fire and smoke
- **Performance warning:** Avoid new/delete on each particle

Area: Random map generation

- Dungeon / outdoors map **procedural generation**
- How to make sure areas are reachable
- Beautification pass to keep game in theme
- Design parameters to meet design intent for each map

Area: Minimaps

- Theory of how minimaps are used in video games (input and output)
- Code should be independent of any entity system
- It should be **interactive** (click on it to navigate) (Ignore Fog of War)
- You should be generating a texture that can be drawn anywhere (UI?)
- Handle circle minimaps, icons and warnings (you are under attack)

Area: Fog of War

- Introduction to all types of Fog of War and why games use them
- Code independent of the game and/or specific entity systems
- First pass with rough edges, *then* smoothing out the edges
- No need for 2D visibility but good to comment it out (check here)

Area: Group movement

- Moving troops in groups via complex terrain
- How do we avoid them lining up?
- How to handle doors? and columns?
- How to finish/cancel the path (avoid all to fill same spot)?

Area: Camera Transitions

- **Fade to Black is too simple!**
- Propose new ways for transitioning
- What if we could have a general manager to switch scenes?
- Taking into account that we no longer activate/deactivate modules
- Transitioning also means moving the camera around with curves

Area: Split Screen

- Creation of N cameras for N viewports
- Loop for render different cameras
- Viewport management
- Performance issues

Area: Sprite ordering and Camera Culling

- Sprite ordering due to Z could be tricky to simulate 3D environments
- We are doing it manually so far, but it must be automatic!
- Also, we do not want to Blit elements outside the camera
- We should only order the entities in the camera

Area: Spatial Audio & Music Manager

- Every entity can potentially generate audio
- Discard entities that are too far
- Check camera position/zoom to play the audio spatially
- Music Manager: Chain a playlist with fading
    - Trigger exploration -> combat music and back

Area: Video Player

- Explain the codec ecosystem and options
  - Check state of the art currently (2020)
- How it works? -> Decompress a streamed buffer of data
- Decompress video frame -> Copy pixel data into texture
- Decompress audio frame -> Feed the samples to audio system

Area: Assets ZIP management with PhysFS

- Deploy your game with all assets compressed in one .ZIP
- **PhysFS** as a filesystem abstraction layer: https://icculus.org/physfs/
- Layers of mounted folders
- Opportunities for encryption

Area: Static code analysis

- Systems for static code analysis to catch more errors in code
- Example for online: coverity, other tools?
- Example for offline: cppcheck, other tools?

Area: Branching Policies

- Explain the concept, the benefits and possible drawbacks
- Git-flow setup and structure for your repositories
- Explain very clearly the use for each case
- How to structure QA around git-flow?
- Relationship with CI tools?

Area: Automated builds CI + CD

- How to config automatic builds with GitHub Actions
- Guide to write scripts, manual copying README files, etc.
- Automatically zip and upload back to github (CD)
- Automatic builds and notifications

Area: Installer

- How to create an installer for your game
- Final touches to be fully integrated with windows:
  - Icons for executable, taskbar app (sizes and formats)
  - Sign the executable with company, windows certification, etc.
- Test the game in win7/win8/win10 with virtual machines
  - Dependencies (libraries and dll needed)