Game Jam Survival Guide

Who‘s that guy?

ROBOVM

@badlogicgames

Preparation? What Preparation?

Choose your tools!

- Engine, framework, library
- Programming language, IDE
- Audio editors & generators
- Graphics editors
- Map editors

Use what you already know!

Let‘s have a look at some engines, frameworks, libraries*

*Ones i‘ve used. Don‘t be religious!

Engines

Pros

- Every platform under the sun
- 2D & 3D
- WYSIWYG editor
- Asset store*
- C#, JavaScript, Boo

Cons

- May be overkill for 2D
- 2D workflow still a bit wonky
- Harder to work efficiently in a team
- HTML5 exports huge/experimental

* Check game jam rules on using 3rd party assets!

Engines

Pros

- Desktop, mobile, HTML5
- 2D & 3D
- WYSIWYG editor
- Blueprints, C++

Cons

- May be overkill for 2D
- 2D workflow not ideal
- HTML5 exports huge/experimental
- C++ if Blueprints isn‘t sufficient

Engines

Pros

- Desktop, mobile, HTML5
- 2D
- WYSIWYG editor
- Game Maker language

Cons

- Custom scripting language
- Can feel very limiting
- Free edition only allows export to Windows

Frameworks

Pros

- Desktop, mobile, HTML5
- 2D & 3D
- Java, Scala, Kotlin, ...
- Documentation
- Very modular & flexible

Cons

- HTML5 export only works for Java
- Users need Java installed for desktop builds
- No official WYSIWYG editor*
- 3D not as powerful as Unity/Unreal
- No asset pipeline, DIY

*Check out Overlap2D!

Frameworks

Pros

- Desktop, mobile, consoles*
- 2D & 3D
- C#, F#
- Derrived from XNA

Cons

- No HTML5 export
- Users need .NET/Mono installed to run desktop builds
- No WYSIWYG editor
- 3D not as powerful as Unity/Unreal
- Asset pipeline can be hinderance

*With some caveats

Frameworks

Pros

- Desktop, mobile*
- 2D
- Lua
- Great API
- Builds native executables

Cons

- No HTML5 export
- No WYSIWYG editor
- Debugging support is not as strong

*Experimental

Libraries

Pros

- Desktop, mobile, HTML5*
- 2D
- C or anything that binds to C
- Minimal API

Cons

- You‘ll have to reinvent a lot of wheels
- No built-in support for common things like tilemaps, fonts etc.
- Performance can be a problem if you don‘t use it with OpenGL

*Via emscripten

How to pick?

Most important

- Do i know it already?
- Do my teammates know it?
- Is it the tool i‘m fastest with?

Bonus

- Does it fit my budget?
- Does it export to HTML5?

If you didn‘t use it before the jam, don‘t use it during the jam!

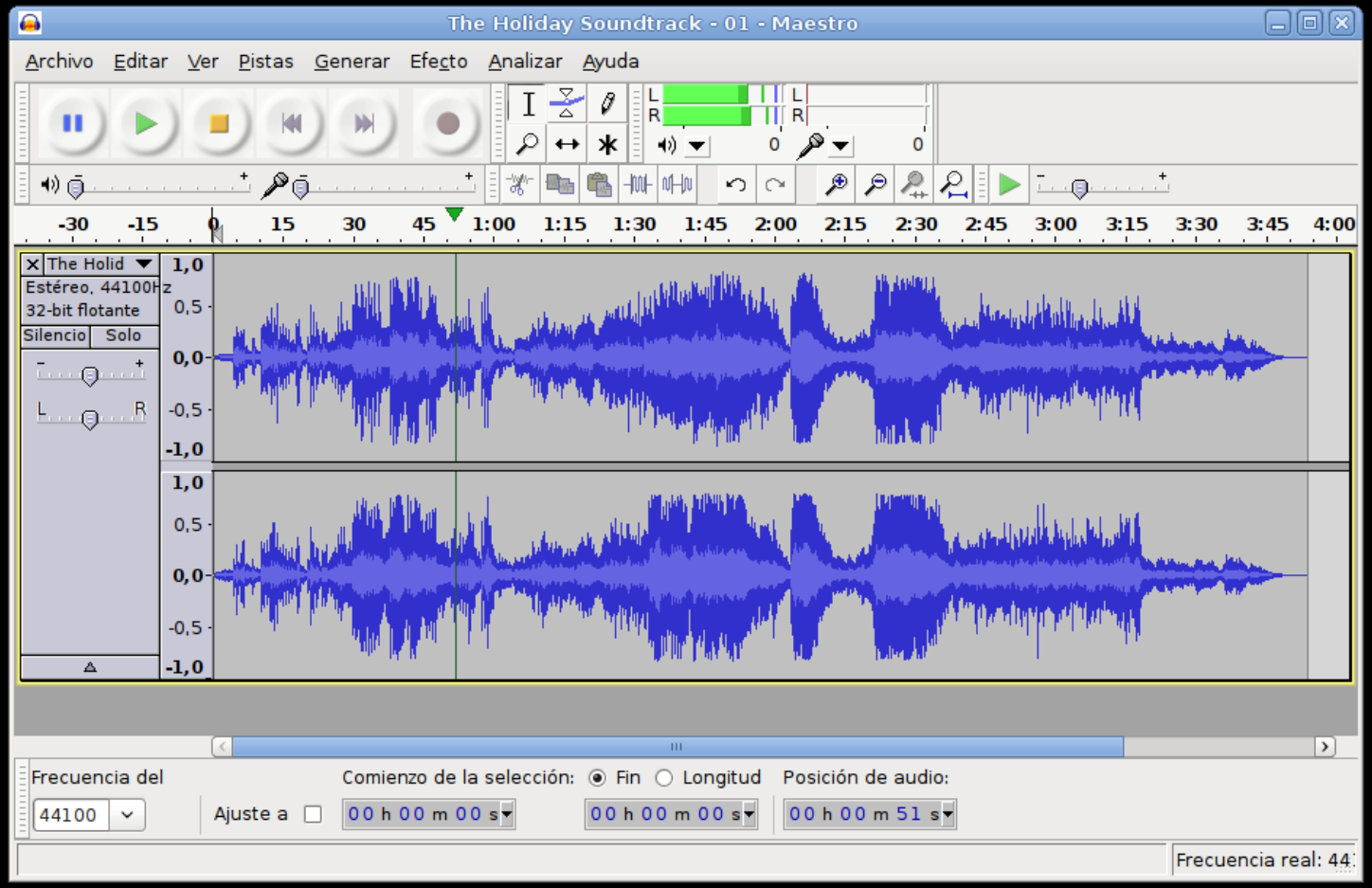
Audio Tools & Resources

http://www.bfxr.net/

Audio Tools & Resources

http://www.audiotool.com

Audio Tools & Resources



http://audacity.sourceforge.net/

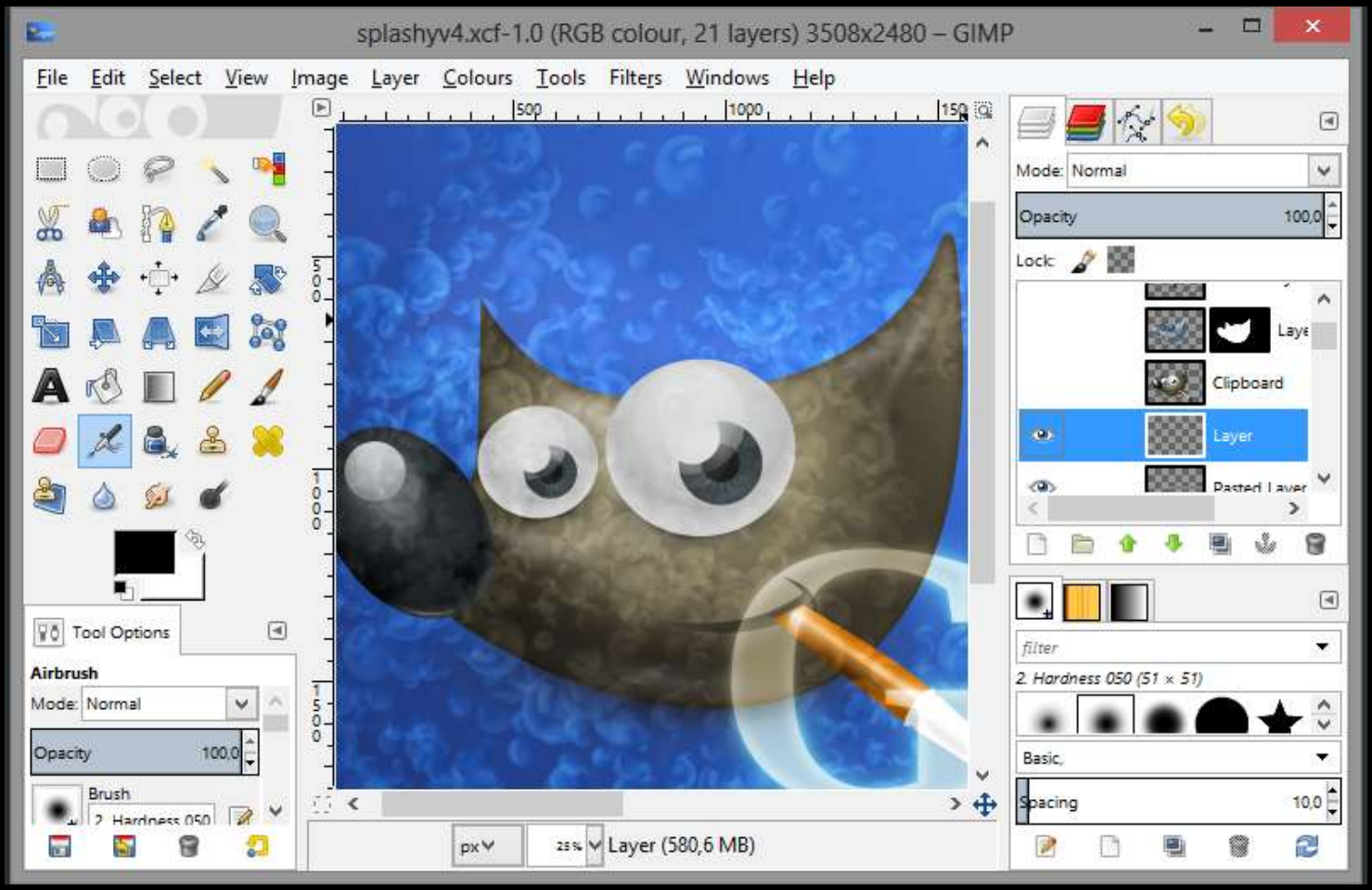
Audio Tools & Resources

Free Soundeffects & Music*

- https://www.freesound.org/
- https://soundcloud.com/ (CC Group)
- http://openmusicarchive.org/
- http://dig.ccmixter.org/
- http://www.indiegamemusic.com/

*Always check Jam rules

Graphics Tools & Resources

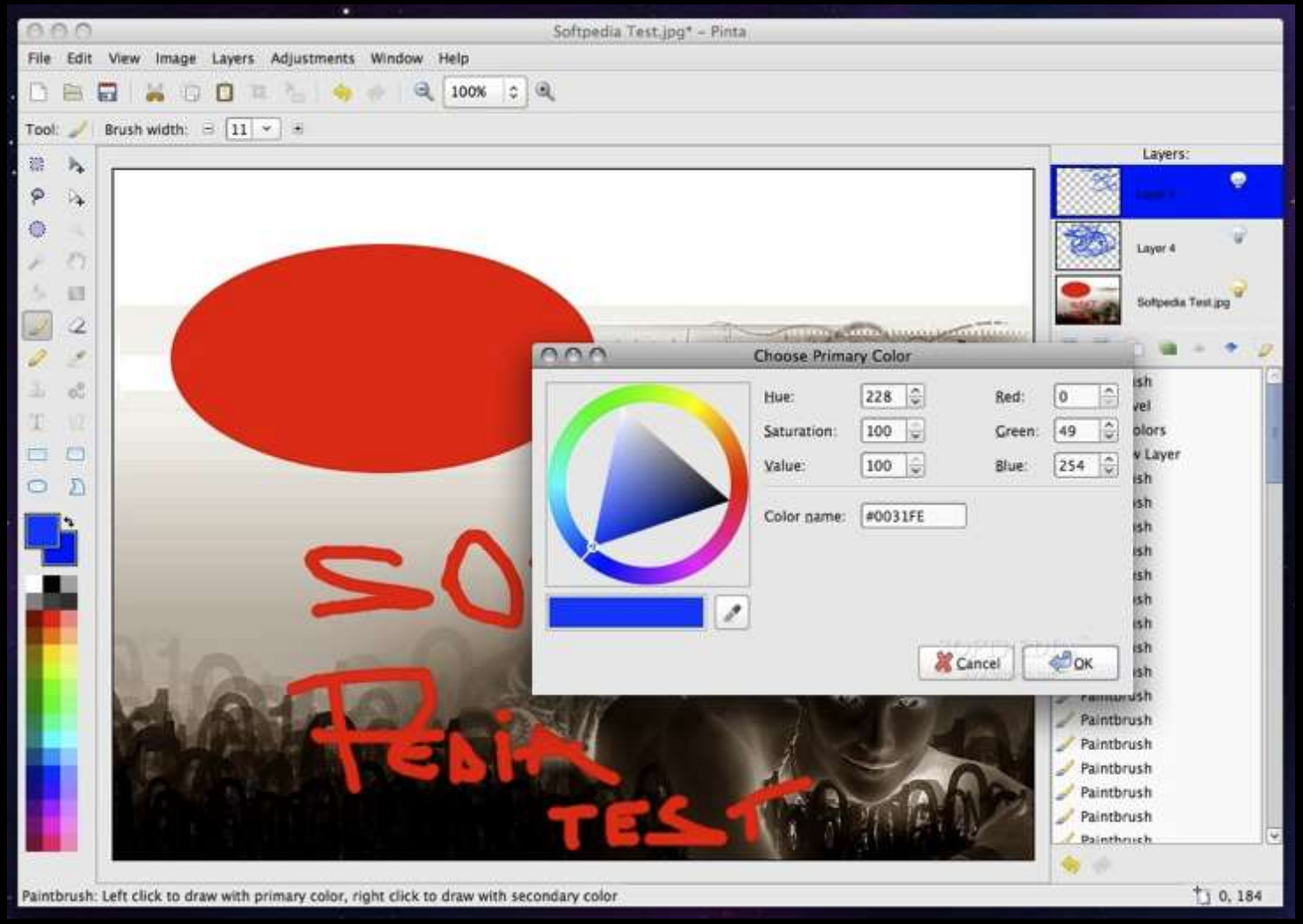


http://www.gimp.org/

Graphics Tools & Resources

http://www.getpaint.net/index.html

Graphics Tools & Resources

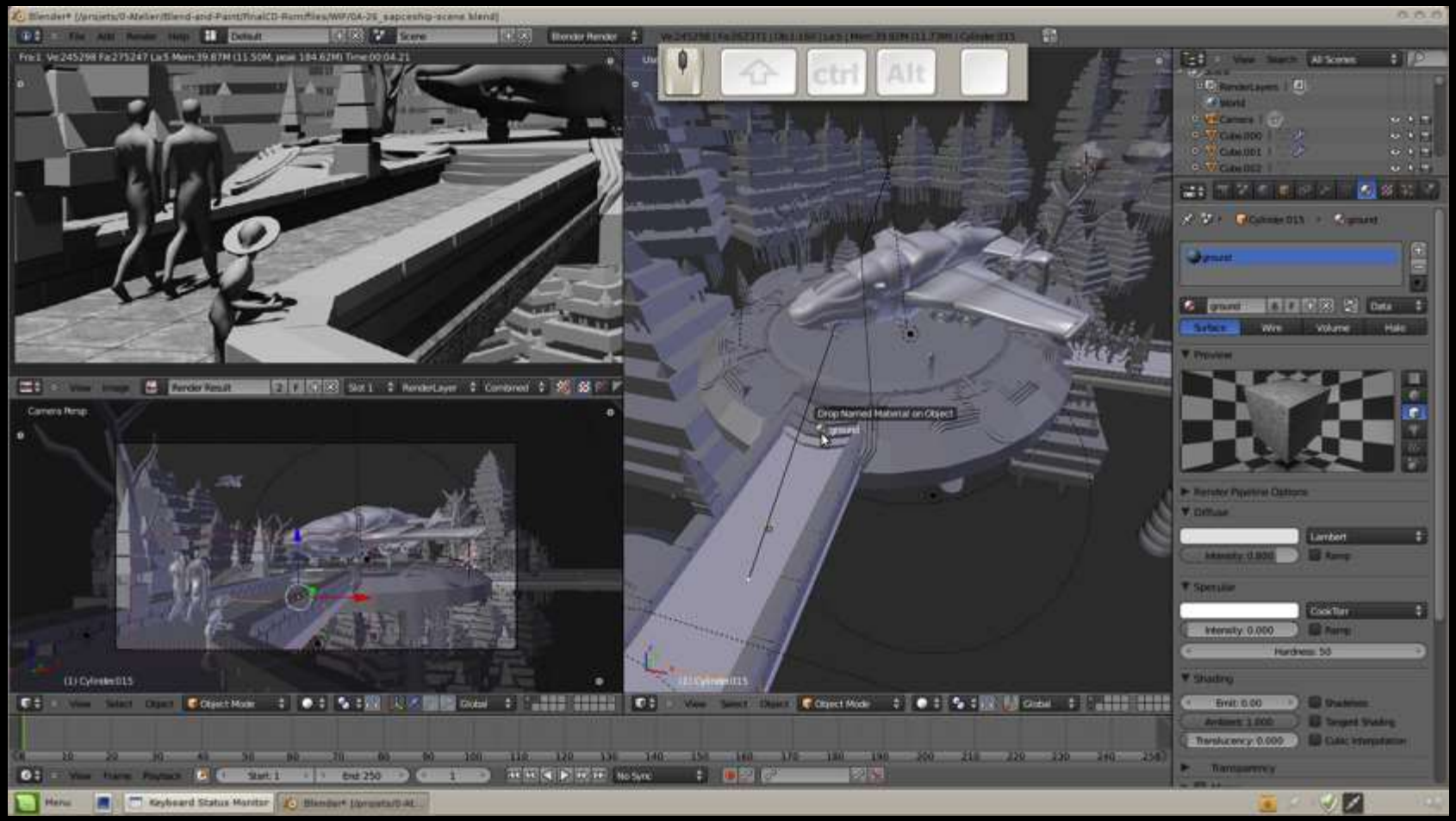


http://pinta-project.com/releases

Graphics Tools & Resources

http://esotericsoftware.com/

Graphics Tools & Resources



http://www.blender3d.org/

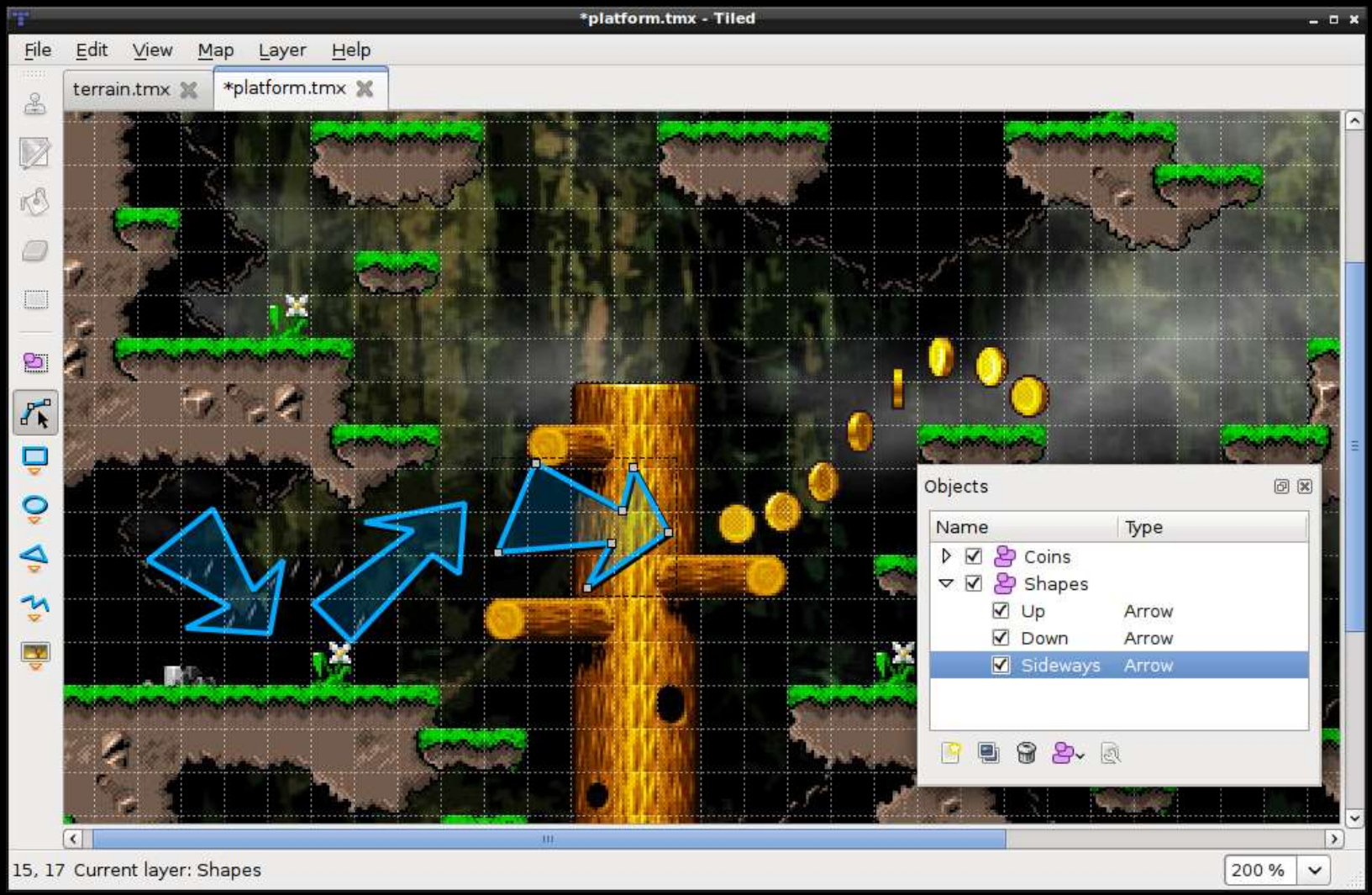
Graphics Tools & Resources

Free Game Art*

- http://opengameart.org
- http://kenney.nl
- http://www.lostgarden.com
- http://www.gameart2d.com

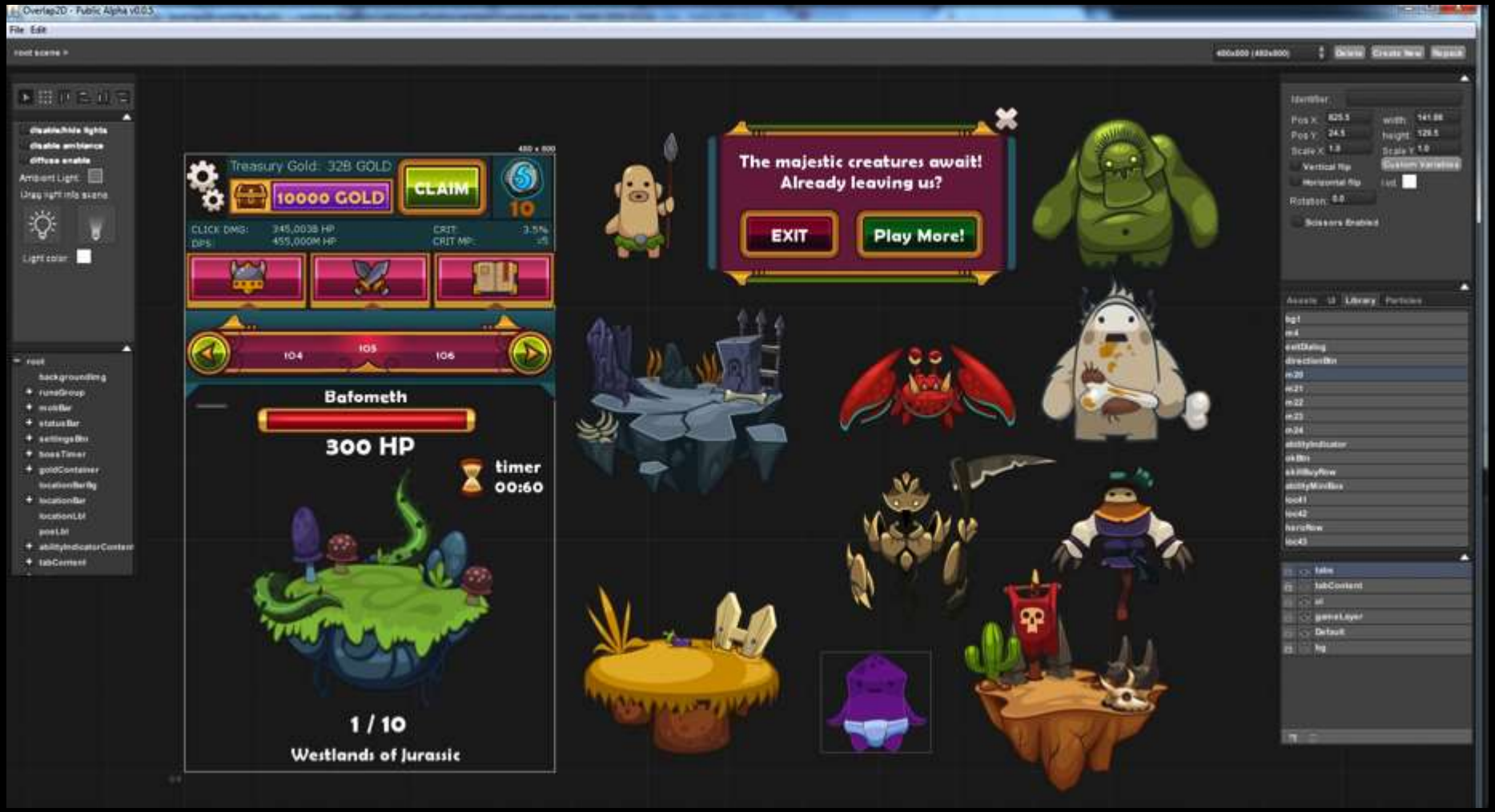
*Always check Jam rules

Map Editors



http://www.mapeditor.org/

Map Editors



http://overlap2d.com/

The 5 Phases of Jamming

1. Team Building
2. Brainstroming
3. Setup
4. Implementation
5. Finishing Touches

Team Building

You will need

- Developers
- Audio & graphics artists*
- Game/level designers
- Coordinator

The latter two can be done by anyone!

*Rare unicorns

Team Building

Developers

- Do the programmy bits
- Need to split up tasks among them
  - Graphics, Controls, Physics, UI, …
- The less overlap code-wise the easier!
- Need to tell artists what formats they need
- Need to define how game/level designer creates content

Team Building

Graphics & Audio Artists

- Do the artsy bits
- Need to split up tasks among them
  - UI, background, characters, effects, ...
- Need to agree on an art style
- May need to create placeholder art early on

Team Building

Game/Level Designer

- Does the content bits
- Needs to define the game mechanics
- Needs to define the game progression
- Needs to create „levels“
- Needs to playtest and give feedback to devs and artists

Team Building

Coordinator

- Makes sure everyone knows what to do
- Keeps track of things to be done
- Keeps track of dependencies between team members
- Keeps track of time
- Keeps track of human needs (food, sleep)

Team Building

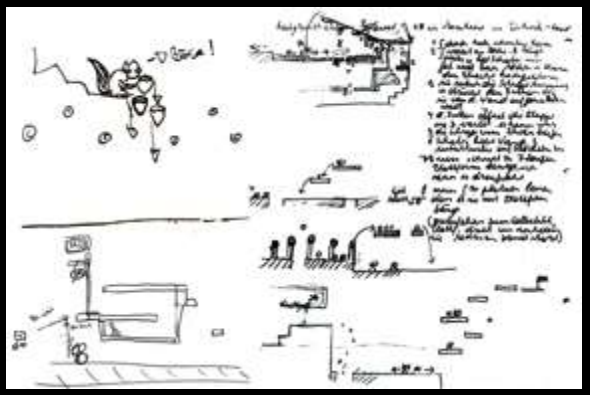
- If you have no developers, look into Construct 2
- If you have no artists, use preexisting art or programmer art
- If you have no game designer, everybody becomes a game designer
- If you have no coordinator, pick one person
- If you are alone, you get to do all the things :D

Brainstorming

Goals

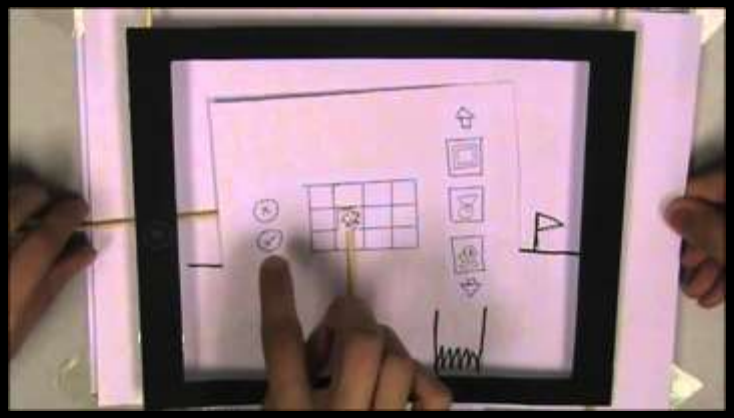
- Get a high-level understanding of your game
  - Genre
  - Game mechanic
  - Setting & Story
  - Art style
- Take time limits into account
  - FPS, MMORGP, RTS are likely not your best bets
- Think outside the box! (hurr durr...)

Brainstorming



To-do

1. Gather ideas from everyone
2. Pick most promising one via vote
3. Define genre & game mechanics
   - Use pen & paper!
4. Define setting & story
5. Define art style
   - Let artists draw quick mockups



Don‘t be afraid to throw things away

Don‘t be afraid to iterate, take your time

Setup

Goals

- Get a detailed understanding of your game
  - What will the developers have to do?
  - What will the artists have to do?
  - What will the game designers have to do?
- Define interfaces between all team members
  - How do developers work with each other?
  - How do artists get their art into the game?
  - How do game designers create game content?
- Define tasks and their order for every team member!
  - Coordinator responsible for keeping track of tasks

Setup

To-do

1. Developers agree on platform & tools to use
2. Artists agree on artstyle
3. Developers and artists agree on how to get art into the game
4. Developers and game designer agree on how to create content
5. Each subteam defines their initial tasks
6. Coordinator keeps track of things

A super lightweight Kanban-like board can help

Implementation

Goals

- Get the damned game done!
- Ensure to have a playable prototype early
  - Prioritize tasks accordingly
  - Game mechanics first to see if they are fun!
- Realize you‘ll likely not get everything done!
  - Which is why you should have something playable at almost all times
  - Cut corners, kill features, focus on the core of your game

Implementation

To-do

1. Every sub-team works on their task
2. Coordinator keeps track of progress
3. Sub-teams talk whenever they need to (re-)define and prioritize (new) tasks
4. Goto 1

Your highest priority should be to have something playable early on!

Implementation

Tips for Developers

- Use source control (git, SVN), do NOT use shared drives, ZIP files, e-mail!
- Don‘t code for re-use
- Don‘t optimize
- Try to create a modular-design so people don‘t depend on each other to much
  - One person responsible for graphics, one for UI, one for AI, one for controlls, etc.
- Make sure game designer can create content as early as possible
- Make sure artists export to easy to use formats
- Make sure artists work for some standard resolution!
- Make sure artists & game designer understand limitations

Implementation

Tips for Artists

- Make it easy to export your art to the proper format
- Make sure everyone uses the same coordinate system/resolution!
- Use descriptive names for files
  - Good: badguy-walk-left.png , Bad: w_1_2.png
- Have one shared folder (Dropbox, Google Drive) containing assets ready for the game designer/developers to integrate
  - Don‘t put multiple versions of the same thing there!
  - Have whatever local folder structure for work in progress assets

Implementation

Tips for Game Designers

- Talk to the developers about what‘s possible and what‘s not
- Focus on simple mechanics but try to put in a twist
- Favor simple level-design over „brainy“ complex levels – they take to long to design!
- If you have down-time help/be the coordinator!

Implementation

Tips for Coordinators

- Ensure that everyone can stay busy
  - Gather the team to discuss new tasks or reprioritize current tasks
- Check on progress regularly
  - If something takes to long, as the team to reprioritize/kill features
- Make sure everybody is reminded they are human
  - Make everyone take breaks
  - Make people go to sleep
  - Make people eat and drink
- If you have down-time, take on a task you can do!

Finishing Touches

Goals

- Submit a playable game before the deadline :D

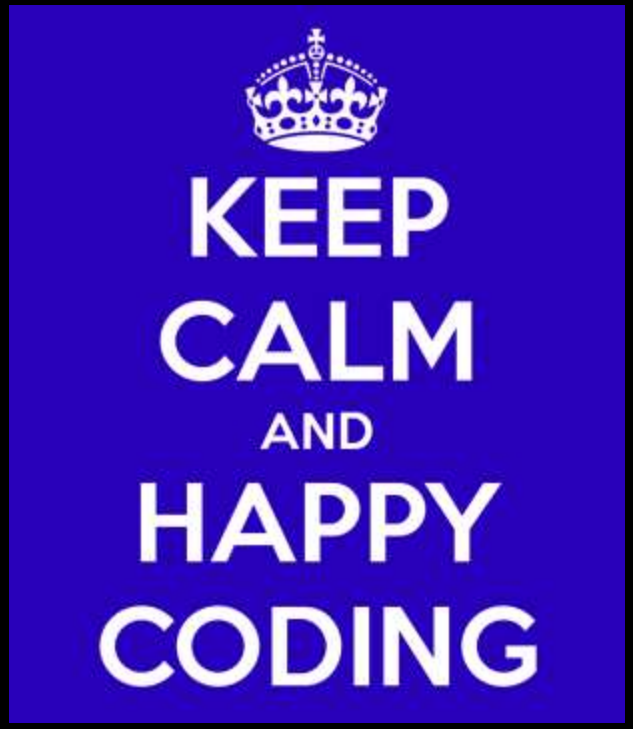
Finishing touches

To-do

1. Feature freeze 2-3 hours before the deadline
2. Create a build for submission
3. Get team together and decide what to polish in the remaining hours
4. If polishing works out, create a new build for submission

Final Thoughts

- Sleep!
- Eat & Drink!
- Take breaks!
- Make new friends!


KEEP
CALM
AND
HAPPY
CODING