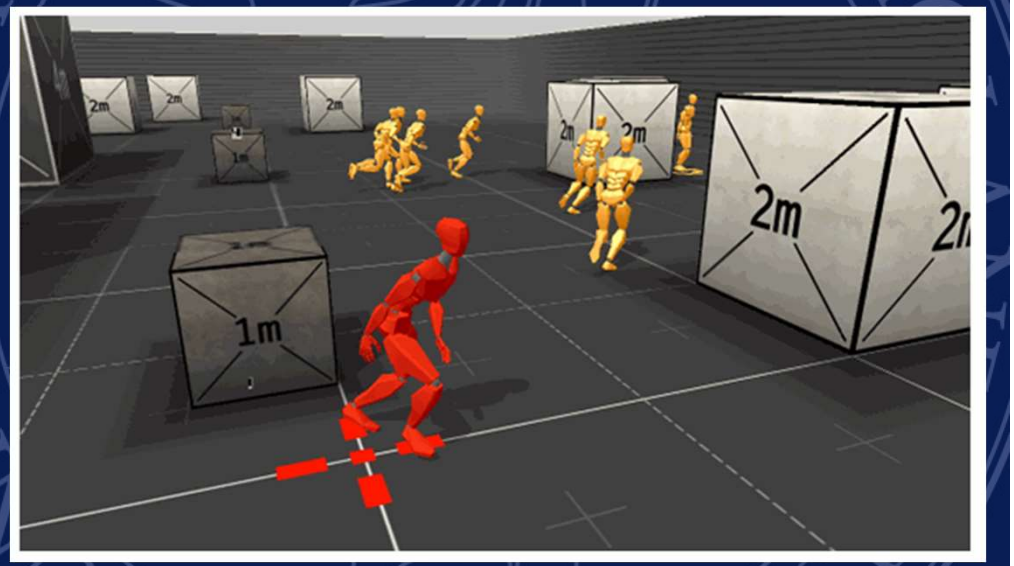
PON.9
Playlab fOr inNovation in Games

UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI INFORMATICA

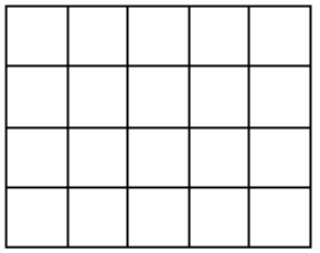
OGD Lesson 009: Prototyping

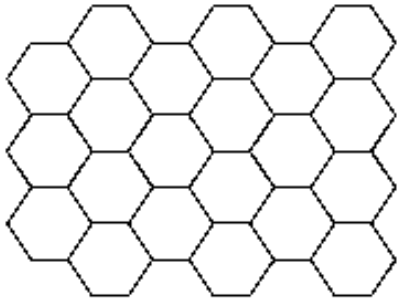
Laura Anna Ripamonti
ay 2021-22



O09. Summary

- What prototyping is good for?
- How to prototype
- References:
  - Chapters 9 , 10, 11 «Game Design Workshop» by T. Fullerton
  - Chapters 8 «The art of Game Design» by J. Schell



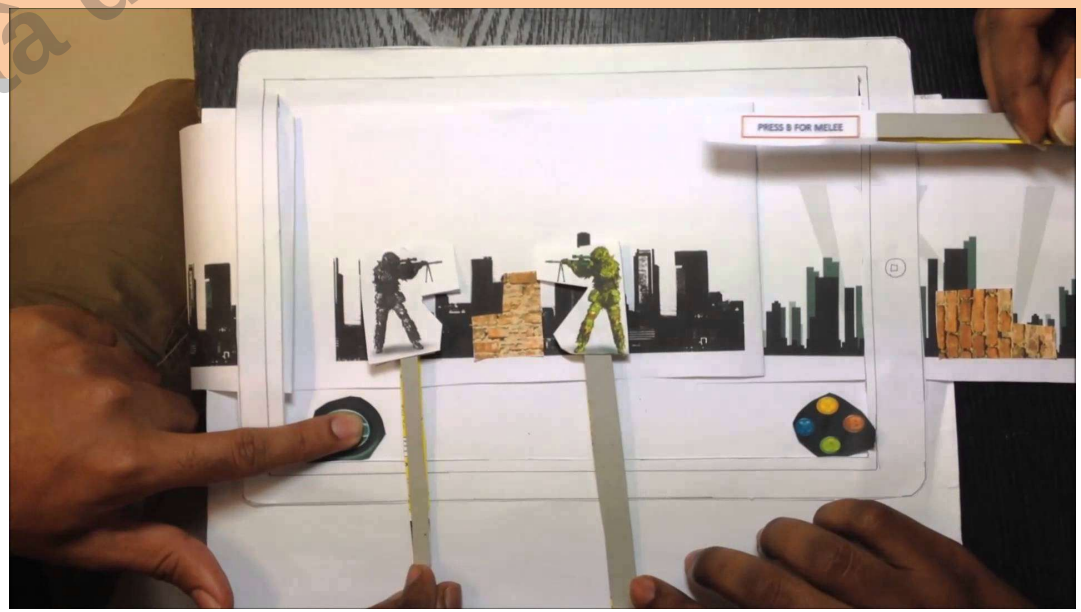


What prototyping is good for?

- Prototyping **is the very heart of game design**: there's nothing more valuable for improving gameplay than good prototyping!

- **Prototyping is the creation of a working model to test feasibility of the game**

- Game proto are **playable version** that may include a little of artwork, sound and features



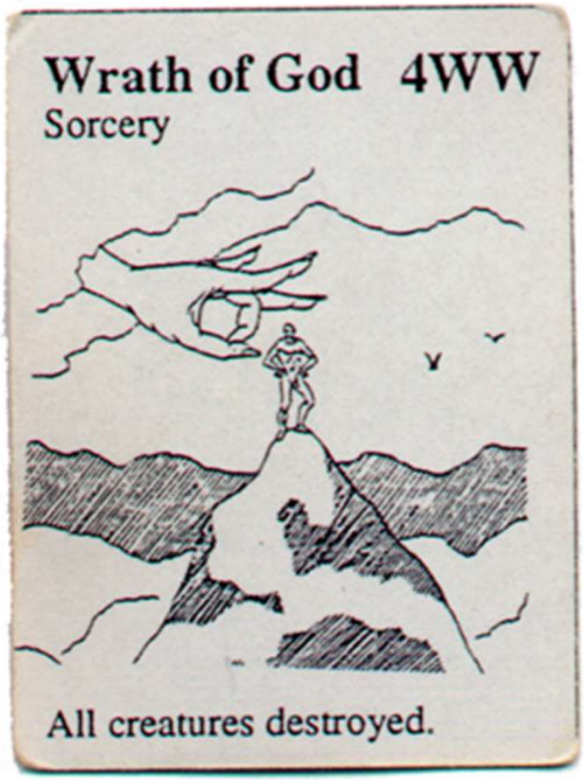
What prototyping is good for?

- The main reason for prototyping is **RISK MITIGATION**
- You need to:
  - Make a **RISK ASSESSMENT** for your game project

How to prototype: methods

- Many **types**: physical, visual, video, etc.
- Generally a single project requires **SEVERAL prototypes**, addressing different aspects/features
- Remember:

prototyping is NOT creating the final design



Tips for productive prototyping

1. Answer a question

- every prototype should be designed **to answer a question, clearly stated**:
  - How many animated characters do the engine supports?
  - Is our core gameplay fun?
  - Etc.

2. Forget quality!!

- All that matters is that the prototype answers the question(s)

Tips for productive prototyping

3. Don't get attached

- The 1° version of your game is **not** going to be a finished product, but a prototype that u'll throw away (it may happen many times)
- Each prototype is only **a learning opportunity**

4. Prioritize your prototypes

- Prioritize the dangers emerged from the risk assessment

Tips for productive prototyping

5. Parallelize prototypes productively

- Have **many small different prototypes** to test different aspects (art, technical aspects, story, etc.)

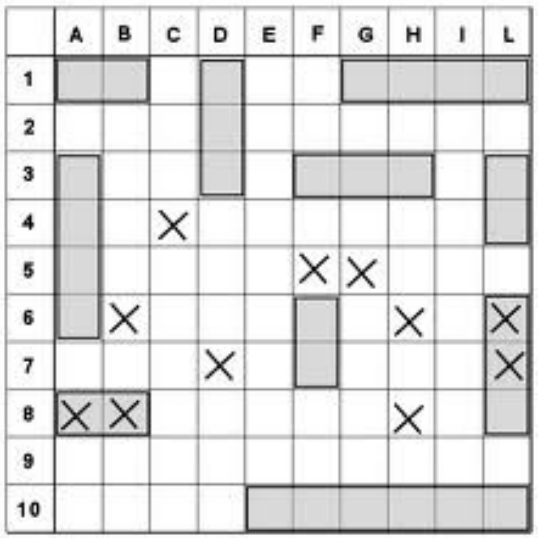
6. It doesn't have to be digital !!!

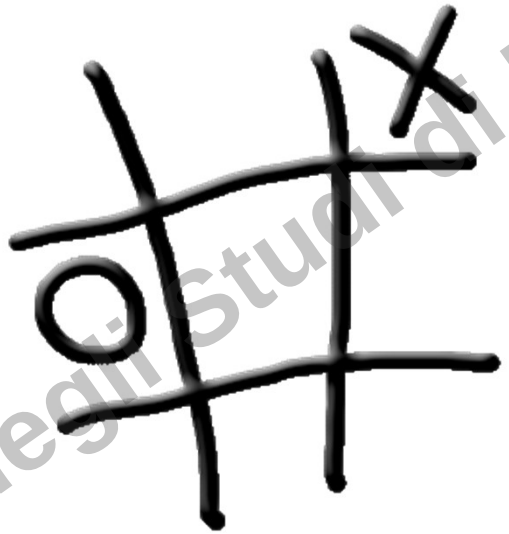
- Cheaper
- Faster
- Allows careful balancing
- Real time games can be paper-prototyped too

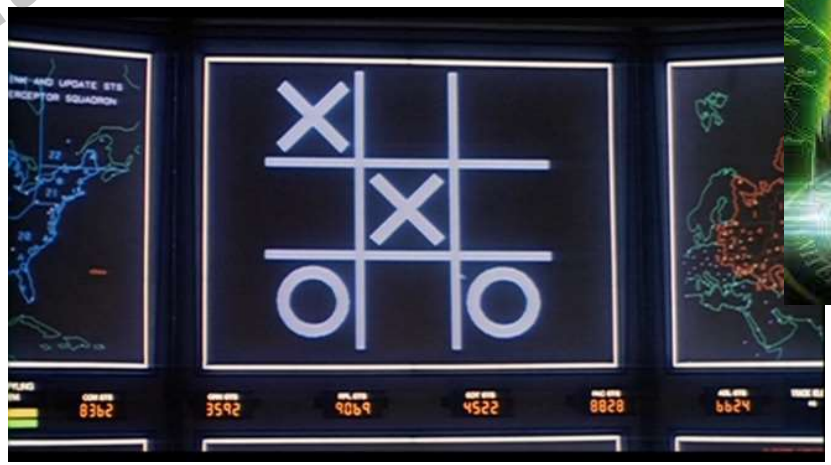
Benefits of physical prototyping

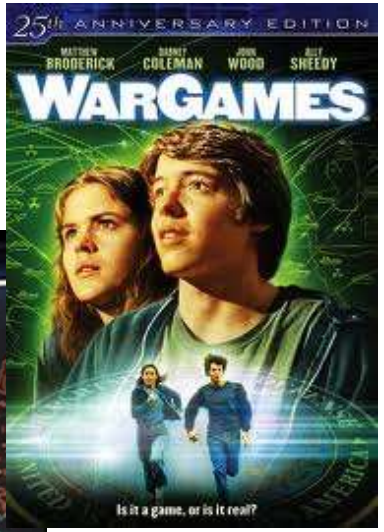
- Focus **on gameplay**, NOT on technology
- Realtime incorporation of playtesters' suggestions
- Deeper sperimentation, since very cheap
- No skills (tech) required => broader participation of the team => more perspectives!
- Fundamental in early stages to test gaming **experience**
- Ensures that players will grasp the essence of the game
- Fundamental for **communicating (also to the team) a complex system …**

Physical proto: examples

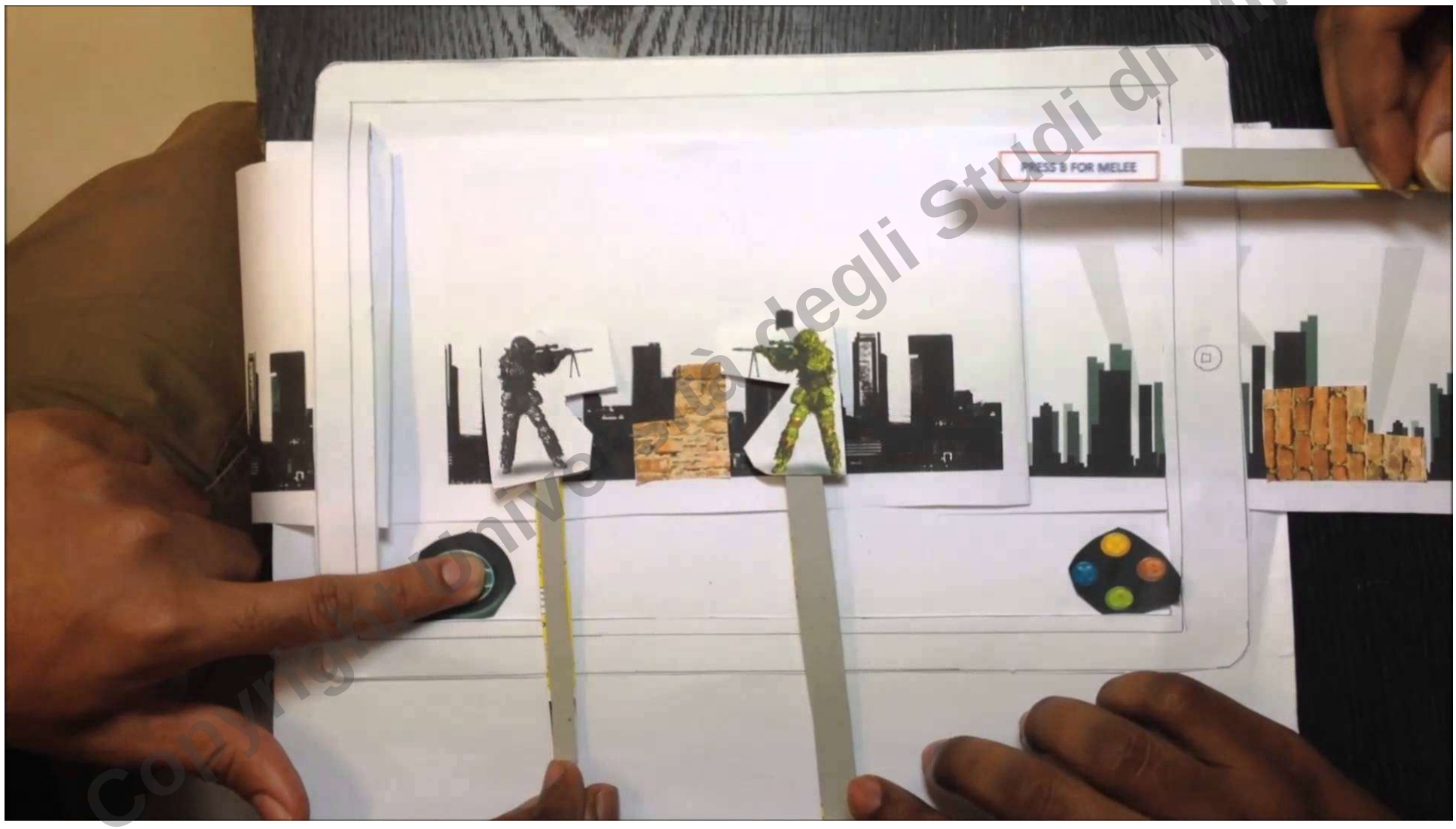






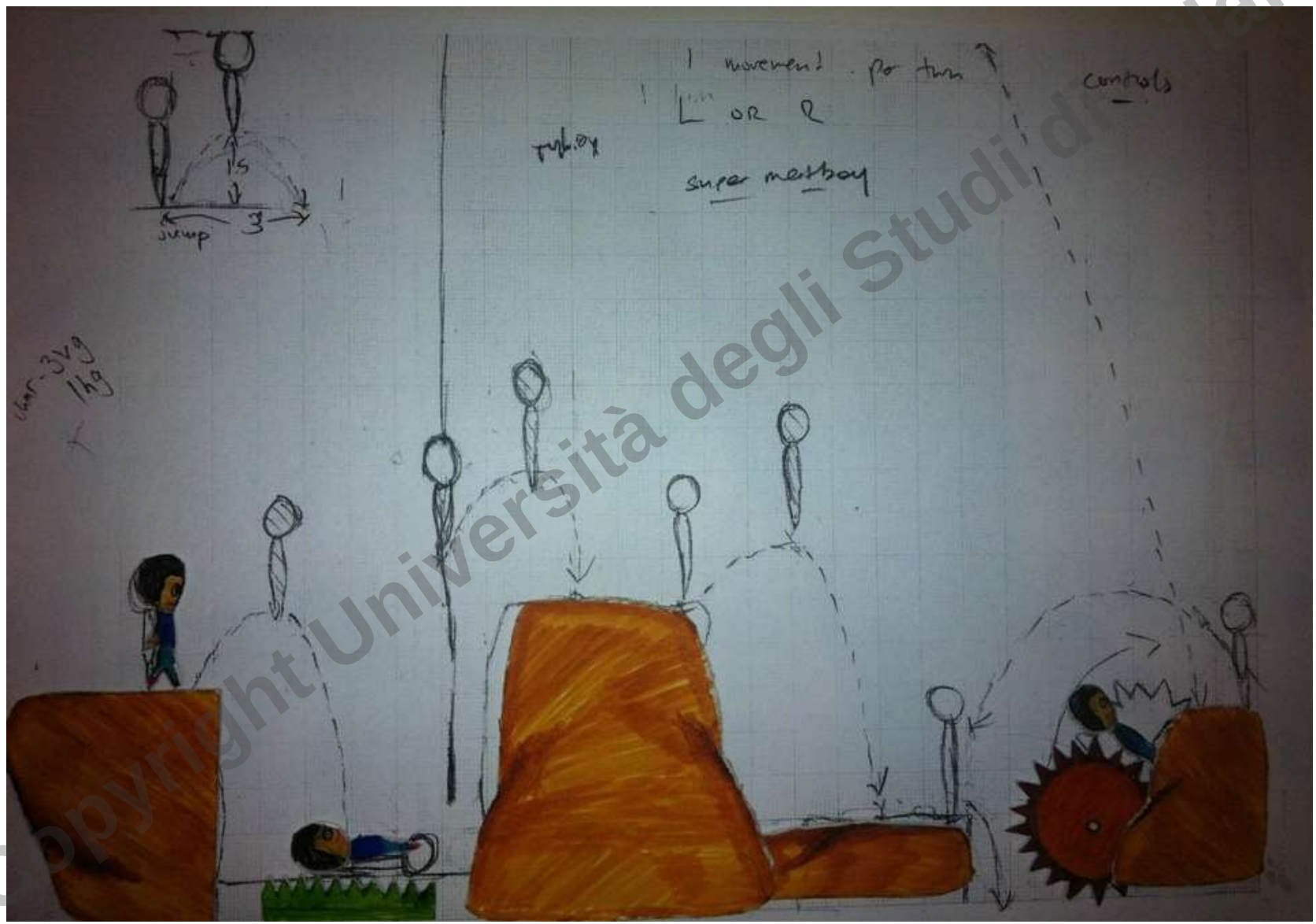


Physical proto: examples



Physical proto: examples

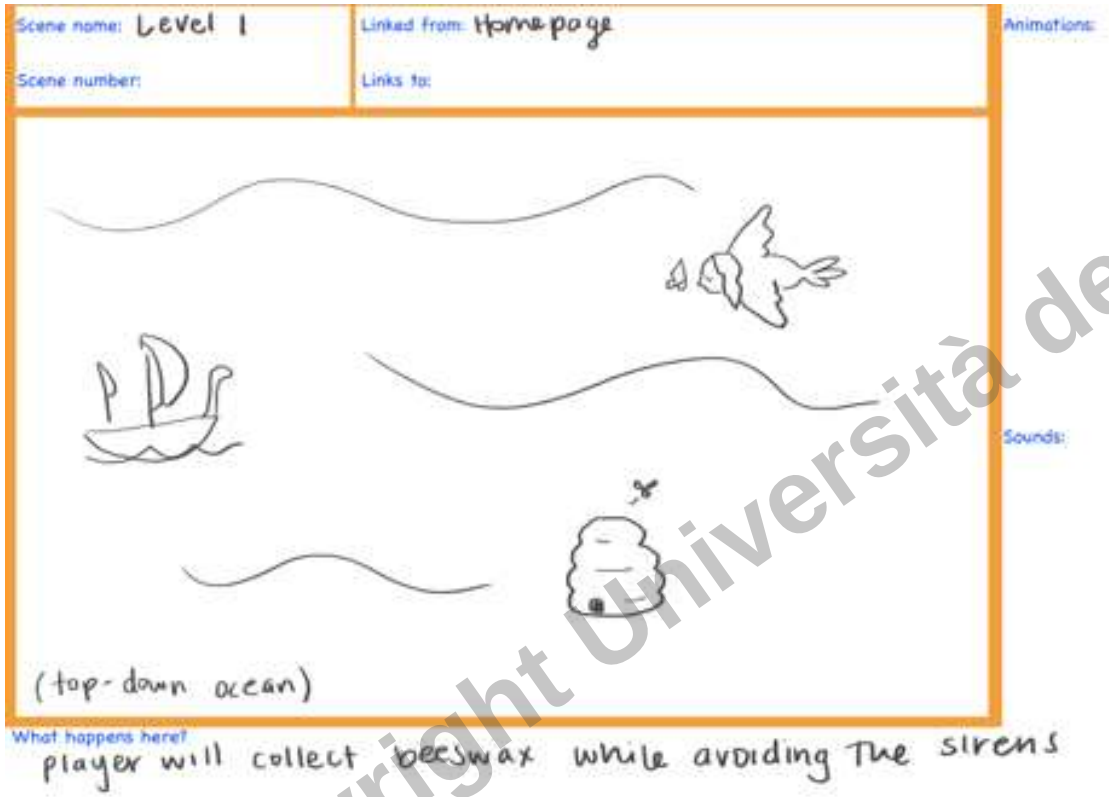
Physical proto: examples

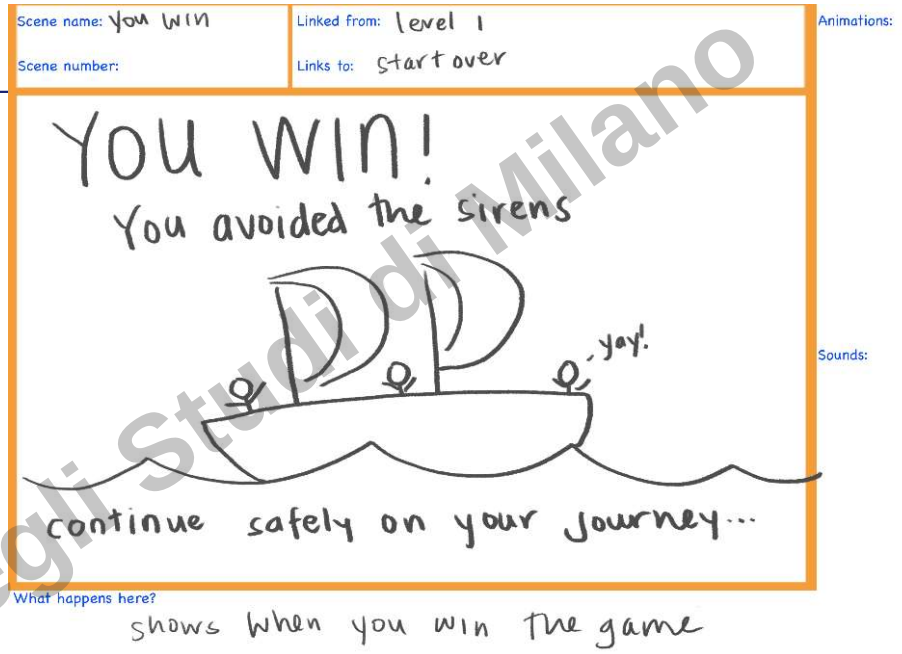


Physical proto: examples

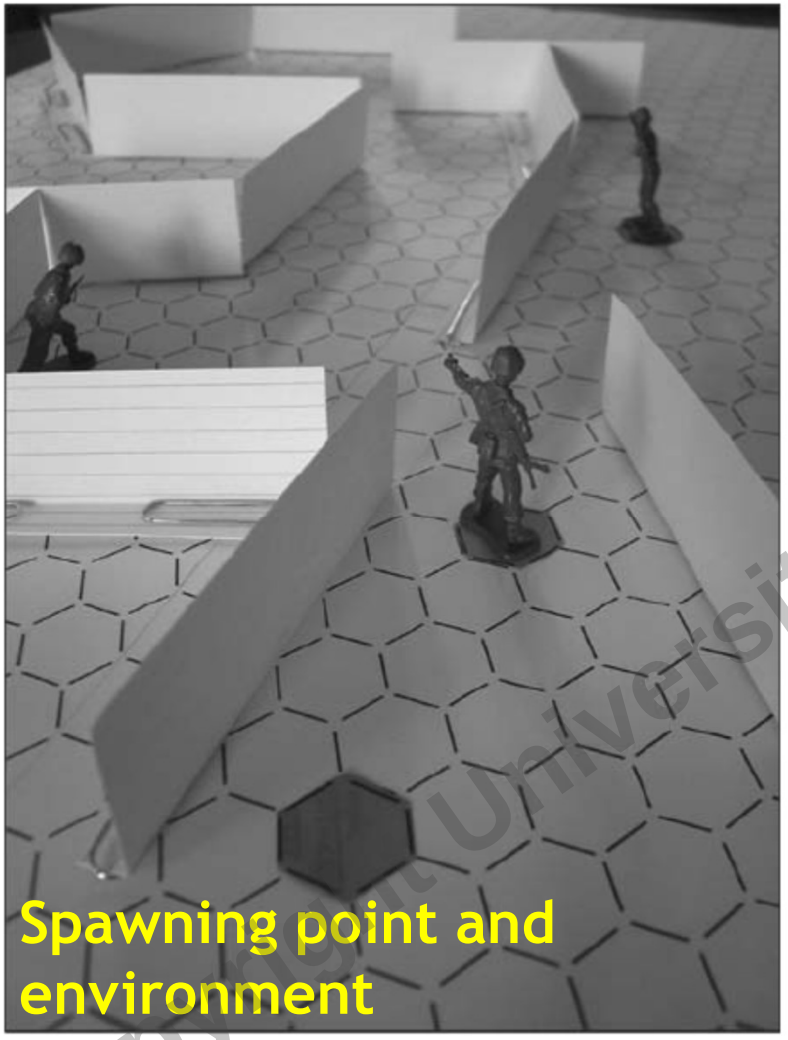
Physical proto: examples

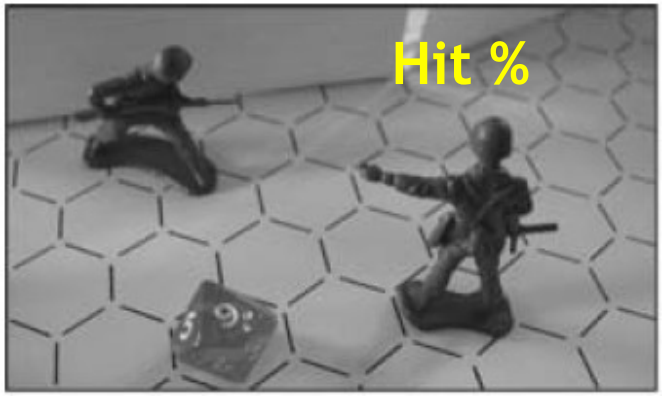
Physical proto: examples





FPS proto





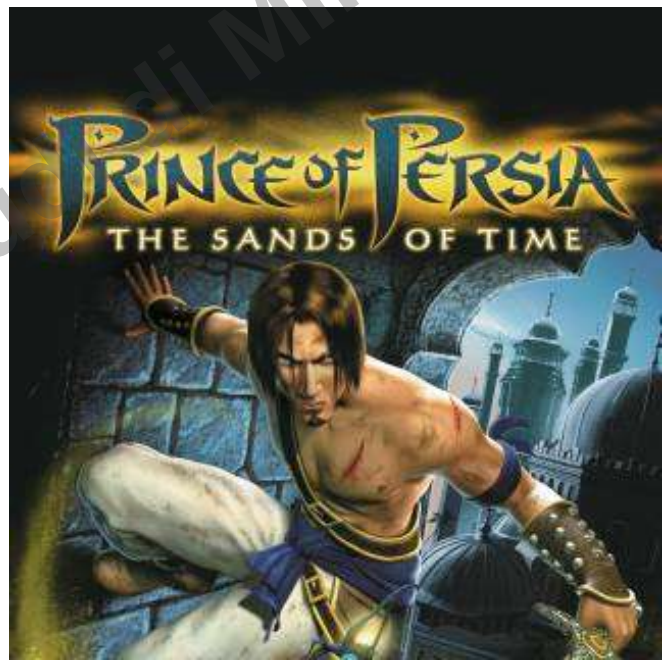
TIPs: search the internet for “hexagon(al) paper” and …
keep adding anything u think may be of use!

Prototyping: physical vs digital

Tips for productive prototyping

7. It doesn't have to be interactive

- Protos do not need to be always interactive
  - Prince of Persia Sands of Time: acrobatics and time reversal have been protyped only with noninteractive animations



8. Pick a «fast loop» game engine

- For the proto choose an engine that allows for recoding your game while running (e.g. Unity)

Tips for productive prototyping

9. Build the toy first

- Many games are build ontop of toys (football, Lemmings, GTA, etc.): be sure the toy is fun to play before buildign the game

10. Seize opportunities for more loops

- Halo was initially a game for Mac, then for PC, then for Xbox … !!!

Tips for productive prototyping: example

- **Problem statement**: create a new type of racing game
- **Solution**: underwater submarine races (with weapons)
- **Risks**:
  - What underwater racetracks look like?
  - Enough innovative?
  - Technology can handle water effects enough?

Tips for productive prototyping: example

- **Prototypes**:
  - Concept art for underwater racetracks

  - GDs building paper proto for testing new mechanics (depth charges, submarines getting out of water and flying, tracking torpedoes, etc.)
  - Programmers testing water effects

Tips for

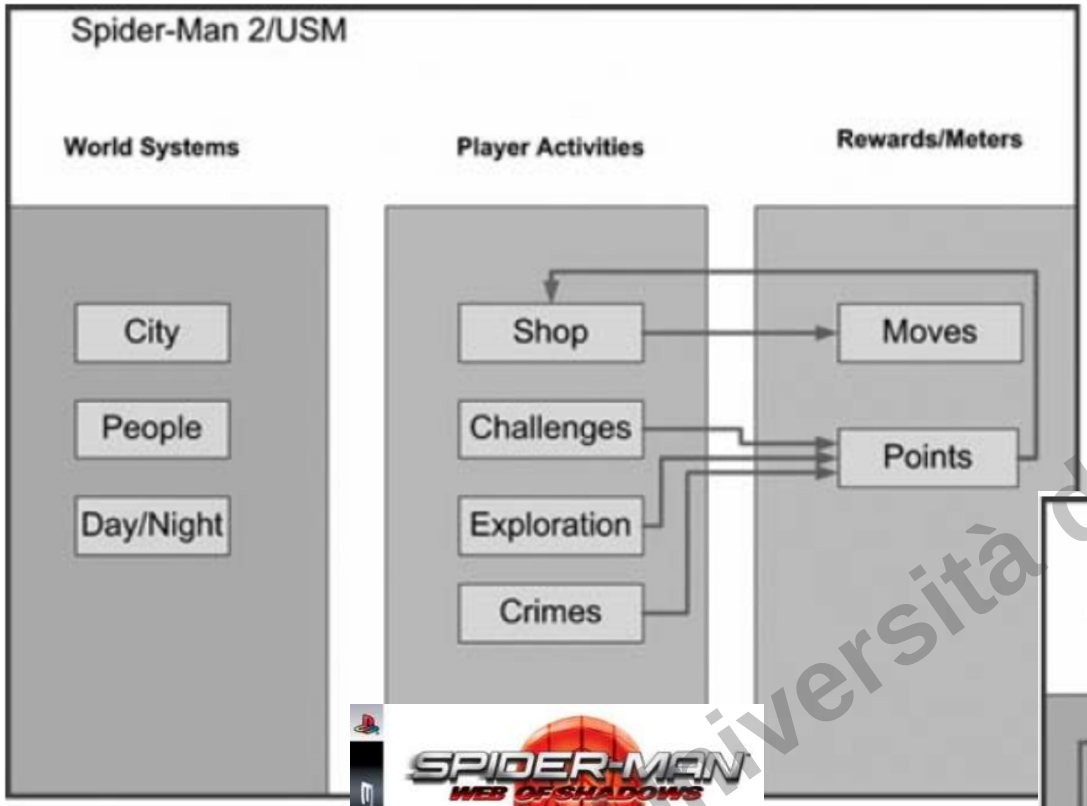
Building a prototype

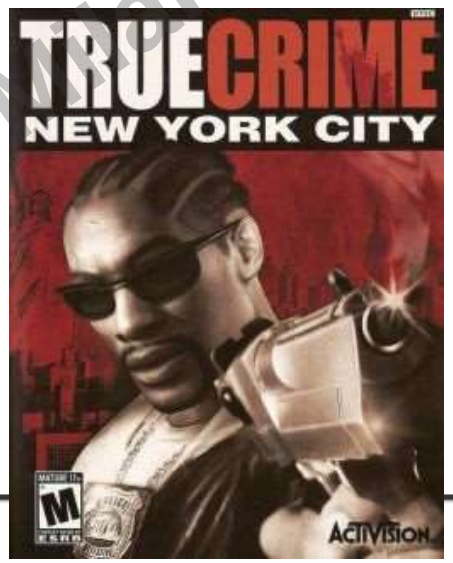
Prototyping your idea

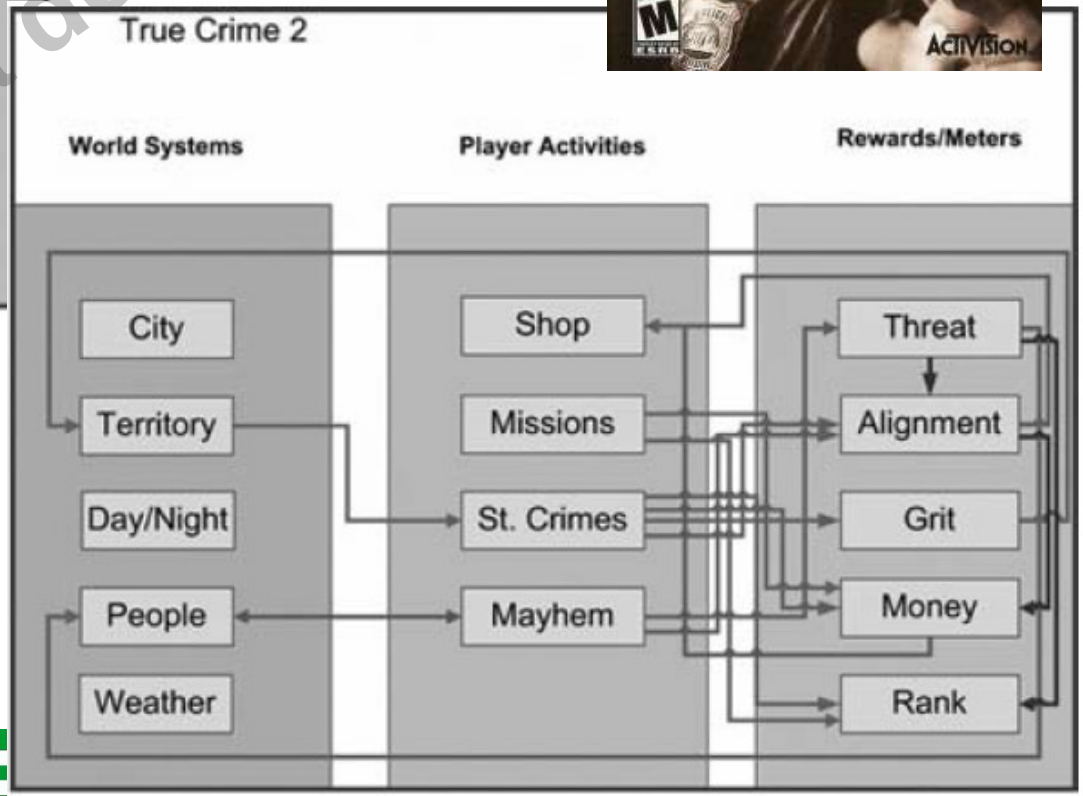
- BEFORE prototyping: make sure to articulate clearly the core gameplay (mechanic)

- CORE MECHANIC(s) = actions a player repeats more often while trying to achieve game overall goal (games are repetitive by nature..)

Core mechanics - examples







Steps to build a physical proto (efficiently)

1. **Foundation** - build representation of core mech: basic game objects (units, resources, …) and key procedures

   - Fewer rules is better
   - Test & change where necessary

Example FPS: movement & shooting rules

Steps to build a physical proto (efficiently)

2. **Structure** – u have the foundation, now you need the framework for the game:
   - Define rules & features essential and the structural elements that support them (FPS: scoring system & hit points)

- **RULES** = modifies to game mech that change how the game functions (ex.: FPS: how to aim and fire)
- **FEATURES** = attributes that only make a game richer (ex.: FPS: more weapons)

=> U can add rules and no features, but you need to change rules to add a feature !

Steps to build a physical proto (efficiently)

3. **Formal details** – on the basis of formal elements, add more rules and procedures to make the game fully functional (ex. FPS: hit %, shields, etc.)
   - Isolate each new rule and test it **individually**
   - At this point the proto is playable!

4. **Refinement** – by trial and errors refine the game adding new features, etc.
   - Rank new features, add it one by one and test individually

BTW: TRUST YOUR TESTERS !

Last but not least …

- A game is a complex system (specific elements could interact in unexpected ways): systematically determine problems and experiment ‘till you solve them!

- For each good idea there is plenty of stupid ones: do not panic! They are the painter’s sketches and help you to learn …
- Physical proto should be used as blueprint for software prototype …

What is digital prototyping good for?

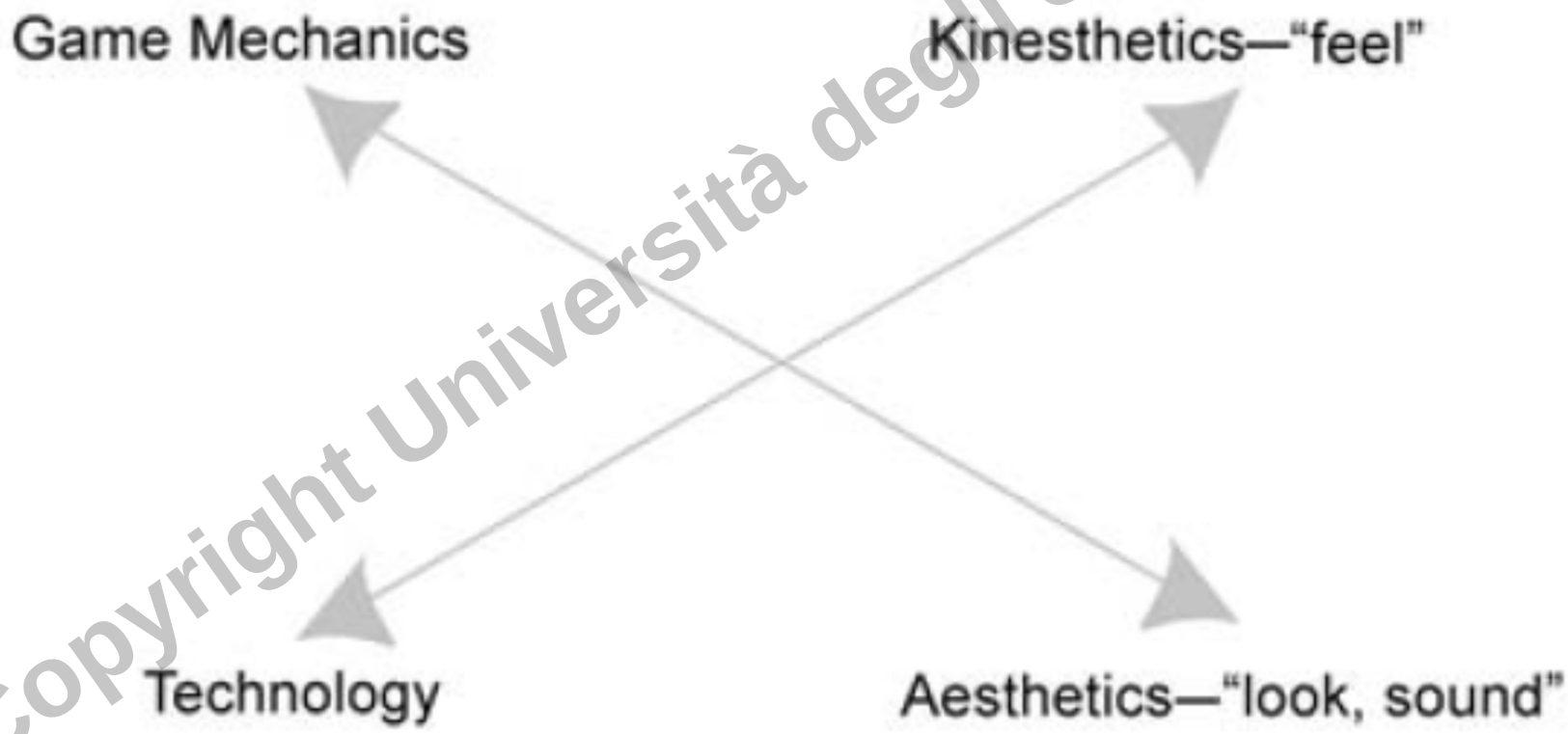
- Physical protos have limits. You'll need also (several) **digital** protos, e.g. to test:
  - Special physics
  - Levels
  - Control system
  - Etc.

First of all …

- Sw proto is OK for understanding and control the elements of the game, NOT for developing part(s) of the game already well known and/or u can proto with cheaper means … !!

Types of sw prototypes

- U'll need SEVERAL protos to test different things …



1. Game mechanics

- It happens that specific (core) gameplay questions cannot be answered by physical prototyping alone
  - Braid: time rewinds
  - Spore: editor

- U need to develop sw prototype of **specific** part(s) of the game!

2. Aesthetics

- Aesthetics = visual & aural dramatic elements (neglected by physical proto)
- Tools for proto:
  - **Storyboards**
  - Concept art
  - Animatic
  - Interface prototype
  - Audio sketches

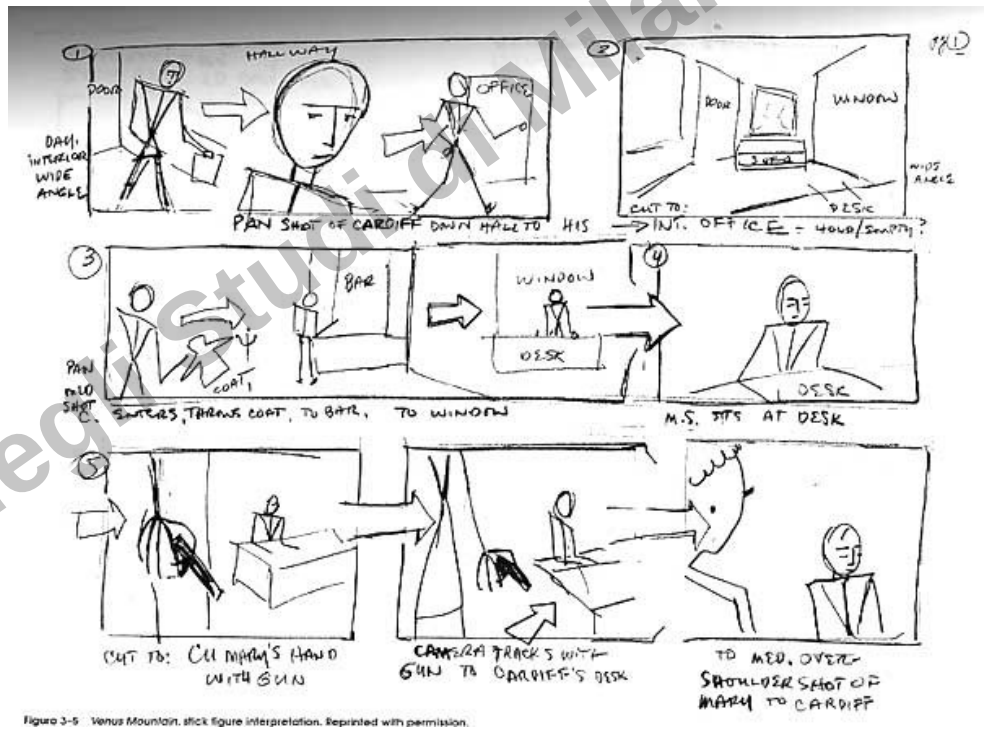
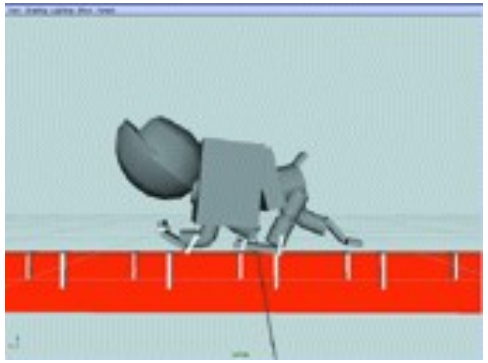


Figure 3-5 Venus Mountain, stick figure interpretation. Reprinted with permission.



http://www.gamasutra.com/view/feature/2899/giving_life_to_ratchet__clank_.php

3. Kinesthetics

- Kinesthetics = the “feel” of the game (how the controls feel, interface responsiveness, …)
  - It **MUST** be prototyped!
  - Linked with the type of controls available

4. Technology

- U need models of the sw necessary for the game to work: graphics, AI, physics, etc.
- This activity is NOT sw engineering, NOR the "real" code, but a quick & dirty way to test ideas
- RAPID PROTOTYPING: small, fast, throwaway projects about specific aspects you want to validate

- TIP: use another language for proto! ;-)

Designing control schemes

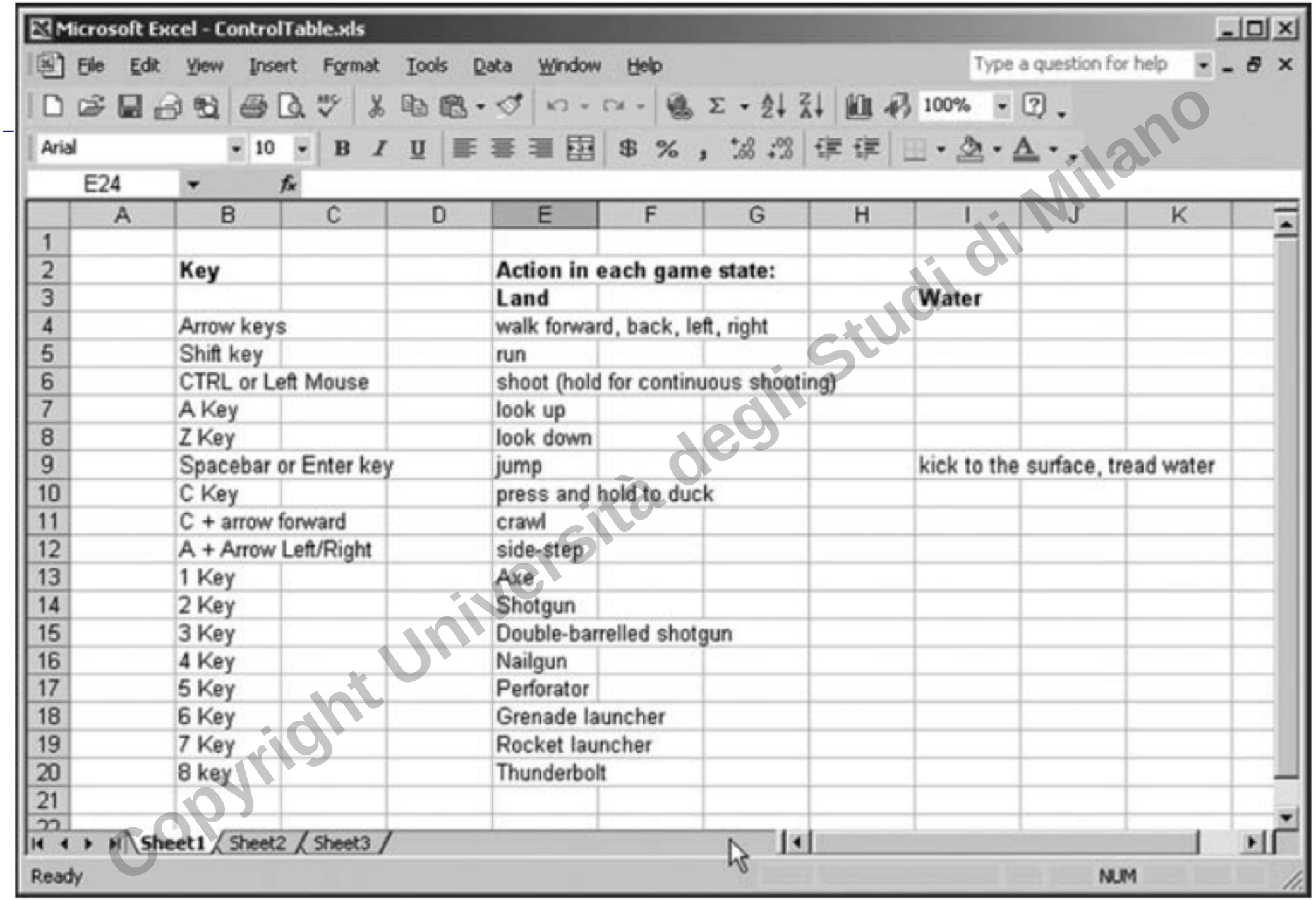
- Key task: design good & INTUITIVE controls
- Most diffused controls are evolution of arrows keys …
- Recent controls: footpad, guitars, Wiimote, Move, Kinect, balanceboard, VR, touch screens …

Designing control schemes (steps)

1. Make sure to understand what the controller(s) of the chosen platform(s) can/cannot support

2. Decide how to best exploit them for your game, on the basis of the procedures defined through the physical prototype!

3. Create a control table (excel is OK!) to keep trace of everything



Microsoft Excel - ControlTable.xls

File Edit View Insert Format Tools Data Window Help

Type a question for help

Arial 10 B I U

E24

	A	B	C	D	E	F	G	H	I	J	K
1											
2		**Key**			**Action in each game state:**						
3					**Land**				**Water**		
4		Arrow keys			walk forward, back, left, right						
5		Shift key			run						
6		CTRL or Left Mouse			shoot (hold for continuous shooting)						
7		A Key			look up						
8		Z Key			look down						
9		Spacebar or Enter key			jump				kick to the surface, tread water		
10		C Key			press and hold to duck						
11		C + arrow forward			crawl						
12		A + Arrow Left/Right			side-step						
13		1 Key			Axe						
14		2 Key			Shotgun						
15		3 Key			Double-barrelled shotgun						
16		4 Key			Nailgun						
17		5 Key			Perforator						
18		6 Key			Grenade launcher						
19		7 Key			Rocket launcher						
20		8 key			Thunderbolt						
21											

Sheet1 Sheet2 Sheet3

Ready NUM

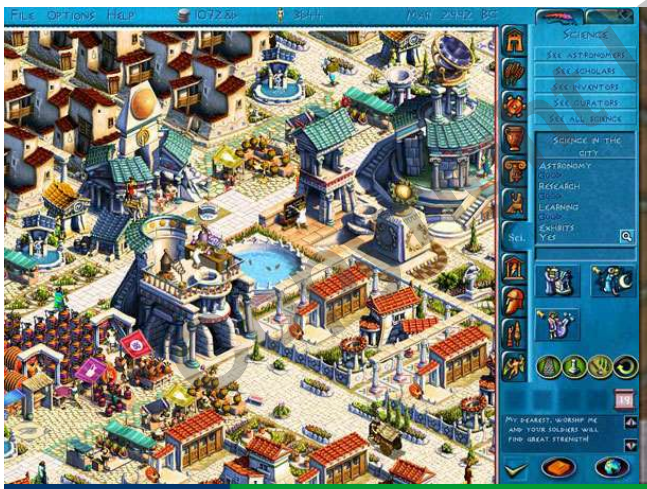
Selecting Viewpoints

- **Overhead view**: diffused among old games (Pac-man), but now adopted mainly for board games & level maps

- **Side view**: popular with arcades/puzzles (Tetris, Donkey Kong), more diffused as side-scroller. Now quite out of favour, but powerful and simple …

Selecting Viewpoints

- **Isometric view**: strategy games, construction sims, RPGs. Ok for “god’s eye”, provides large amounts of info to player. Can be 3D
- **1° person view**: Limits player knowledge, OK for adding dramatic moments
- **3° person view**: adventure and sports game, OK for more detailed character control

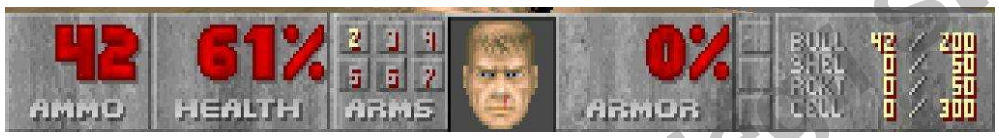


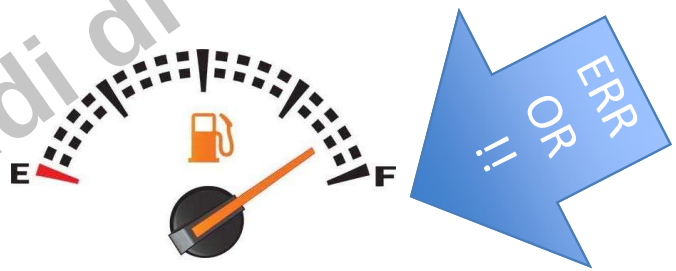
Interface desing

1. **Form follows functions**: the design of an object must come from its **purpose**
   - Ask yourself which are the formal els, BEFORE designing interfaces/ controls
2. **Metaphors**: take the dry facts of a PC memory and display them in a way that fits the game experience
   - Consider the **BASIC METAPHOR** of the game: which metaphor better suits formal & dramatic elements (RPG inventory = backpack, etc.)?
   - Beware of respecting the **MENTAL MODEL**

Interface desing

- **Visualization**: help players to visualize large amounts of info at a glance
  - Beware of **NATURAL MAPPING**





- **Grouping features**: grouping similar features (weapons, movements, etc.) togheter will help players to find them
- **Consistency**: when changin' scene, features should NOT change position
- **Feedback**: aural/visual/tactil/sound feedback should ALWAYS let the player know her action has been received

Prototyping tools: game engines

www.unrealengine.com

- Unreal is for free
- … well … it's Unreal
- A good occasion to learn C++ (highly prized in the idustry)

(free, with constraints)

http://unity.com

- "market standard"

(free & open source)

www.panda3d.org

- Python and C++

(free & open source)

http://www.ogre3d.org

- Direct3D 9 & 11, Metal, OpenGL (incl. ES2, ES3 and OGL3+) and WebGL (Emscripten) support
- Windows, Linux, Mac OSX, Android and iOS support
- Builds on various compilers such as MSVC, GCC 4.8+ or Clang

Prototyping tools: other

- SCRATCH

http://scratch.mit.edu/

. - Developed by MIT and free

- makes it easy to create interactive stories, animations, games, music, art

- stencyl

www.stencyl.com

  - intuitive toolset derived from Scratch, free
  - publishes for iOS (iPhone/iPad), Android, Windows, Mac, Linux, HTML5 with “no coding”
  - takUsed also in some big studios

Prototyping tools: game engines

- RPG MAKER (free trial version) www.rpgmakerweb.com/
  - Easy creation of RPG with “no programming”
- Adventure Game Studio (free)
  - Win, no Mac
  - Easy creation of adventure games

www.adventuregamestudio.co.uk/

- THE GAMES FACTORY 2 by ClickTeam
  - Ok for Arcade games, Platform games, Adventures, with “no programming”

CLICK TEAM

www.clickteam.com/the-games-factory-2

Prototyping tools: game engines

by GarageGames

- Free: MIT open source but: you need to pay for additional packages (e.g. art, etc.)
- Ok Win, no Mac
- Ok multiplayer
- Torquescript (similar to C++)
- Engine written in C++ (in case u'd need to edit it …)

http://www.garagegames.com/products/torque-3d

Prototyping tools: other

- **Level editors**:
  - WarCraft & StarCraft map & unit editors
  - Aurora Toolset (Neverwinter Nights)
  - Minecraft (why not?)

- **Programming languages**:
  - C++, C# (Object Oriented => ok for large scale development teams)
  - Java, ActionScript, Visual Basic, Python, Processing

Any other thing u think may be of use ;-)