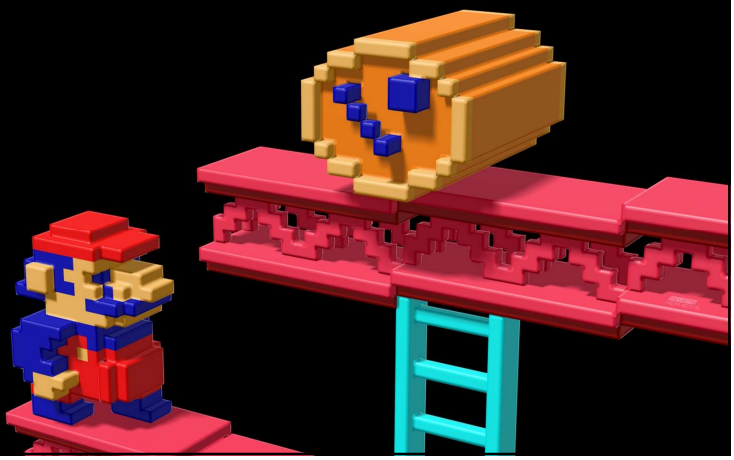
# 3D Video Games

Marco Tarini
Università degli Studi di Milano
2023/2024



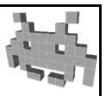
1

## 3D Video Games

Marco Tarini
Università degli Studi di Milano
2023/2024

- Core techniques used in modern 3D games
- A well-established set
  of specific methodologies
  used in most 3D games

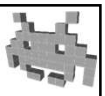
2



## Game Categories: according to gameplay

- Puzzle game
  - Color matching
  - Hidden object
  - Trivia game …
- Action game
  - Beat'em up
  - hack'n'slash
  - Fighting
  - Pinball
  - Platform
  - Maze
  - Shooter
    - FPS
    - MMO FPS
    - LightGun
    - Shoot'em up (shumps)
    - Rail shooter
    - 3rd person
- Action-Adventure
  - Stealth
  - Survival horror
  - Exploration
  - PoP / Tombrider
- Adventures
  - IF - Interactive Fiction
  - Real time 3D adv
  - Point and click
- Board game
  - Card game …
- Strategy
  - 4X
  - RTS
  - Strategy MOBA / MMOG
  - Action-RTS
  - Tower defences
- Vehicle simulation
  - Driving simulator
  - Flight simulator
    - Amateur
    - Combat
    - Space …
  - Racing game
  - Vehicular combat
- Role-playing games
  - RPG (eastern, western)
  - Sandbox RPG
  - MMOPRG
  - Roguelikes
  - Action RPG
- Sport games
  - Soccer / Football / …
- Simulation / management

4



## Categories: according to player types

casual games vs hard core games

5

## Categories: according to platforms

- Arcade
- PC stand-alones
  - Aka “desktop app”
  - Win, Mac, Linux…
- Console
  - Wii, PS, XBox …
- Browser: game = web app
  - html5, webGL, unity, flash…
- Mobile devices
  - Android, iDevices, PSP …

6

## Categories: according to developer

Independent games

- No/tiny publisher:

Mainstream games

- Big publisher

7

## What does a video-game publisher do?

- fund developments
  - including licences
- distribution
- marketing
  - ads, launch, market surveys...
- packaging, manuals
- localization

*High risk!*

ACCOLADE namco Broderbund

8

## Categories: according to developer

**Independent games**

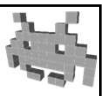
- No/small publisher
- Low starting $
- Small Dev-Teams
- + freedom +novelty
  - (traditionally)
- In need of alternatives for:
  - Funding
    e.g.: Crowd funding
    - see indiegogo.com, kickstarters.com, ...
  - Distribution
    - e.g.: steam, popcap, apple store...

**Mainstream games**

- Big publisher
- Big $ per project
  - (at times, mega-$'s)
- High quality: a must
- Large Dev-teams

9

## Categories: 2D or 3D?

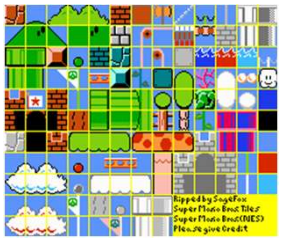


2D games

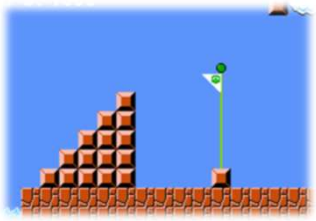
- Sprites + Tilemap

3D games

- 3D Models + 3D Scenes



TileSet

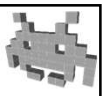


TileMap

Sprites

10

## Categories: 2D or 3D?

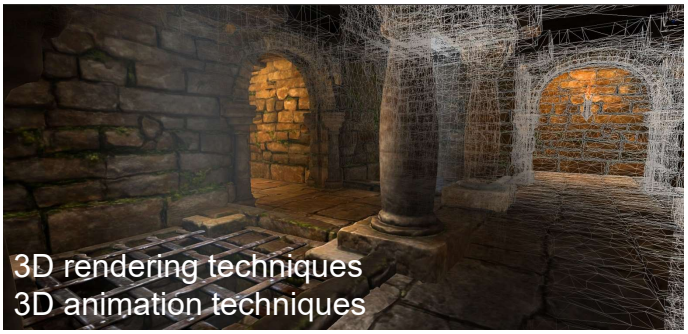


2D games

- Sprites + Tilemap

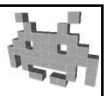
3D games

- 3D Models + 3D Scenes




11

## Categories: 2D or 3D?



### 2D games

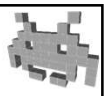
- Sprites + Tilemap
- Techniques:
  - Blitting
  - Tilemaps
    - and 2D scrolling
  - Sprite support
    - sprite collision-detection
    - 2D transform
  - (2D physical engines)

### 3D games

- 3D models + 3D Scenes
- Techniques :
  - 3D Modelling
    - Scenegraph, models
  - 3D Real time rendering
    - 3D transform
    - lighting
  - 3D animations
    - Kinematics, motion capture, model animations…
  - 3D phyisical simulations
  - 3D sound localization

12

## Categories: 2D or 3D?



### 2D games

- Sprites + Tilemap
- Tools:

HTML5, Flash, Allegro, NEO GEO, GameMaker: Studio, …

### 3D games

- 3D Models + 3D Scenes
- Tools:

OpenGL, Microsoft DirectX, OpenGL ES, blender game engine, WebGL, Unreal Engine, unity, gameplay, OGRE, Urho3D, …

13

## Note: we are interested in the tech not the gameplay

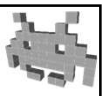

14

## 3D Video Games: fun facts

- Huge industry
- Video games = killer apps
- Technology impulse (HW e SW)
- Performance *and* complexity

16

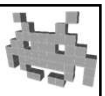
## About this course: webpage

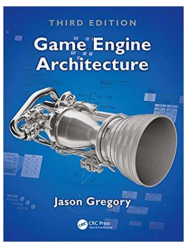


- Follow the link from Ariel
- or
  - Search for my name: Marco Tarini
  - Land on my unimi page
  - Follow 3D Videogame link
- or
  https://tarini.di.unimi.it/teaching/3DVG2024/

18

## About this course: Potentially useful textbooks

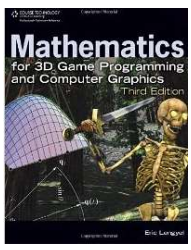





### Game Engine Architecture

Jason Gregory

Complete (notes on:
software tools, software eng., AI prog, CG prog, math, game design...)



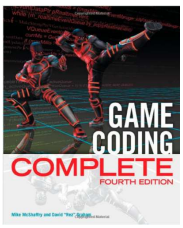

### Mathematics for 3D Game Programming and C.G. (3rd ed)

Eric Lengyel

Good coverage of 3D math,
(and, CG pipeline, geometry + transforms, raytracing, visibility, physic sims, semplice geom processing...)

19

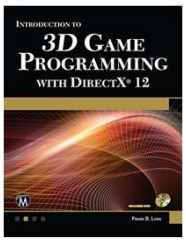
## Other relevant books




Game Coding Complete (4th ed)

Mike McShaffry,David Graham

Practical approach
(sometimes not fully up to date)
Stress on coding asoect, software eng
(e.g. memory managment).



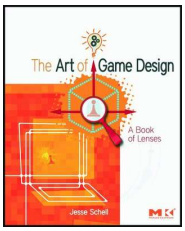

Introduction to 3D Game Programming with DirectX 12

Frank Luna

Rendering / GPU
(basically, Computer Graphics for games)

The Art of Game Design

Jesse Schell
not technical, focus on design!




20

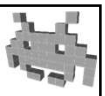
## About this course: the "game of the week"


After every Monday lecture
(including today)

- Completely optional
- Not part of grading
  - No extra point
- Not an official part of the course
  in any sense or form
- Just an occasion to have fun

21

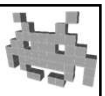
## About this course: the exam



- Preliminary Written Test
  - Moodle
  - Closed and short open questions
  - Mini-problems
  - Definitions.
- Oral Exam
  - Covers the entire lectures
  - Procedure: I roll a die, 1-24
    Ask about respective lecture

22

## 3D Video Games



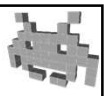

Battlezone – Atari 1980

40 y

Unreal Engine V – 2020

26

## Course Plan

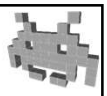


2h

lec. 1: **Introduction** ●
lec. 2: **Mathematics** for 3D Games ●●●●●
lec. 3: **Scene Graph** ●
lec. 4: Game **3D Physics** ●●◖ + ◗●
lec. 5: Game **Particle Systems** ◖
lec. 6: Game **3D Models** ◗●
lec. 7: Game **Textures** ●●
lec. 9: Game **Materials** ◖
lec. 8: Game **3D Animations** ◗●●
lec. 10: **Networking** for 3D Games ●
lec. 11: **3D Audio** for 3D Games ●
lec. 12: **Rendering Techniques** for 3D Games ●
lec. 13: **Artificial Intelligence** for 3D Games ●

29

## Course Plan



☆ = 1 CFU (8h)

lec. 1: **Introduction** ●
lec. 2: **Mathematics** for 3D Games ●●●●●
lec. 3: **Scene Graph** ●

★★ bases

lec. 4: Game **3D Physics** ●●◖ + ◗●
lec. 5: Game **Particle Systems** ◖

lec. 6: Game **3D Models** ◗●
lec. 7: Game **Textures** ●●
lec. 9: Game **Materials** ◖

★ appearance

lec. 8: Game **3D Animations** ◗●●

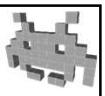
★★ computer animation

lec. 10: **Networking** for 3D Games ●
lec. 11: **3D Audio** for 3D Games ●
lec. 12: **Rendering Techniques** for 3D Games ●
lec. 13: **Artificial Intelligence** for 3D Games ●

★ "bridge" lectures

30

## About this course: Little Extra

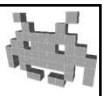


- C++ Coding
  3D Math
  - After Monday lecture
- Bring your own laptop
  - Learn C++
  - Learn math for 3D basics

- Completely optional
- Not part of grading
  - No extra point
- Not an official part of the course
  in any sense or form
- Just an occasion to have fun

31

## About this course: Personal projects



- There is no project in this course

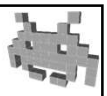
…but…

- You are encouraged to experiment with a game engine
- If you do have an ongoing project which is a 3D Videogame, maybe share your experience!

- Completely optional
- Not part of grading
  - No extra point
- Not an official part of the course
  in any sense or form
- Just an occasion to have fun

32

## Game Engine: tasks



**GRAPHICS**

**PHYSICS**

**ARTIFICIAL INTELLIGENCE**

**SOUND**

**SCRIPTING**

**NETWORKING**

**GUI + INTERFACES**

**ASSET MANAGEMENT**

**...**

35

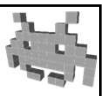
## Game Engine

- A game SW suite which deals with a set of common tasks:
  - Handling of the 3D Scene
  - Renderer
    - Real time transform + lighting
    - Models, materials …
  - Physics engine
    - (soft real-time) Newtonian physical simulations
    - Collision detection + response
  - Networking
    - e.g., LAN via UTP…
  - 3D Sound-rendering, Sound mixer
  - Handling of input devices
  - Main event loop, timers, windows manager…
  - Memory management
  - AI module
    - Common solutions to many common AI sub-problem, e.g., routing
  - Localization support
  - Running scripts
  - GUI (e.g., via interactive HUD elements)

Animations
scripted or computed

37

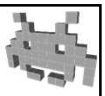
## Implement once, use many times



- Still possible to make games completely from scratch (zero reuse), but increasingly rare.
  - Even many projects/series started this way then switch to a game engine
- Game-engines take care of many common functionalities needed by different games.
  - eg: unity, UNREAL ENGINE, GameMaker: Studio, Adventure Game Studio
- But
  - Reuse = constraints
  - Zero reuse → maximal freedom

39

## Engines which we will *occasionally* refer or adopt for demonstration



unity OR UNREAL ENGINE

41

## Game Dev-Teams

Technical Staff

Designers

Digital Artists

42

## Game Dev-Teams

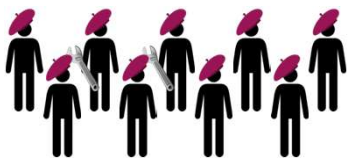
Technical Staff

develop
customize
integrate

game *tools*

game *engine*

Digital Artists



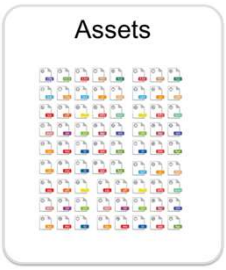
43

# Game Dev-Teams

Technical Staff

Digital Artists



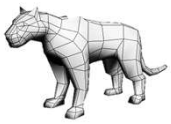



45

# Game assets

## (*aka* game contents)

- 3D data
  - models
  - textures
  - materials
  - shaders
  - animations
  - collision objects
  - scenes
  - *etc*



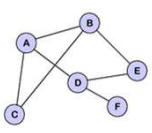
- audio
  - music
  - sound fxs
  - ambient sounds
  - voice overs
  - *etc*

- video
  - cut-scenes, intros, *etc*

- 2D art
  - screen splashes
  - backgrounds
  - GUI / HUD elements
  - [ sprites & tile-sets ? ]
  - fonts
  - *etc*

- text
  - dialogues trees
  - messages
  - translations
  - *etc*

- etc:
  - scripts
  - stats
  - levels
  - *etc*

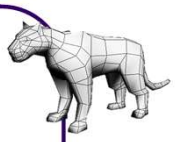



46

## Game assets

### (*aka* game contents)

- 3D data
  - models
  - textures
  - materials
  - shaders
  - animations
  - collision objects
  - scenes
  - *etc*



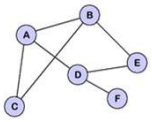
- audio
  - music
  - sound fxs
  - ambient sounds
  - voice overs
  - *etc*

- video
  - baked cut-scenes, intros, *etc*

- 2D art
  - screen splashes
  - backgrounds
  - GUI / HUD elements
  - [ sprites & tile-sets ? ]
  - fonts
  - *etc*

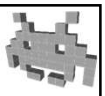
- text
  - dialogues trees
  - messages
  - translations
  - *etc*

- etc:
  - scripts
  - stats / tables
  - levels [& tile-maps?]
  - *etc*




47

## Next lecture…



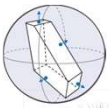
- *Info: from this point on,*
  *the slides cover material that will be presented*
  *in NEXT lecture*

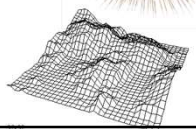
48

# The 3D part of game assets

- 3D Models
  *i.e. tri-meshes with:*
  - per vertex attrib
    - normals, color, AO, ...
  - LODs
  - "uv-mapping"
  - keyframes
    - cyclic animations
    - face-morphs, ...
  - "skinning"
- Materials
  - lighting model stats / flags
  - textures
    - RGB maps
    - normal maps
    - alpha maps ...
  - shaders
    - vertex, fragments, ...

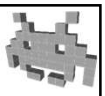
- Animations
  - blend shapes
  - skeletal animations
  - kinematic animations
  - geometry caches
  - skeletons (rigs)
- Geometric proxies
  - hit-boxes
  - bounding objects
  - AI-meshes
- Particle effects
- Environments
  - scene-graphs
  - skydomes
  - 2.5D terrains





49

GENERAL TERMINOLOGY



# A general concept

## we will be encountering it again and again

### ASSET - STORED

- Build during the dev of a game
  - « it is designed »
  - « it is hand-modelled »

👍 quality (usually)
- (if artists are good)

👍 artistic control
- by the digital artist

- costs space (RAM, DISK...)

VS

### PROCEDURAL - GENERATED

- Produced during game execution on-demand
  - « it's a procedure »
  - « it's dynamically generated »

👍 variations
- which is linked to *replayability*

👍 flexibility
- auto-adapts to the context

- costs computation time (CPU, GPU...)

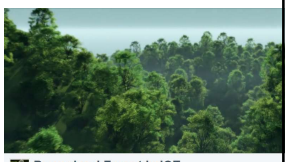
50

# Procedural generation
## In games

For example

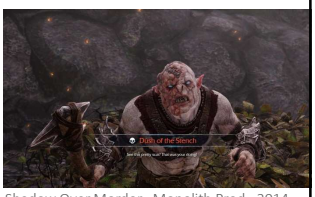
- Procedural levels / missions
- Procedural Terrain
- Procedural AI
- Procedural «Bosses»
- Procedural Scenes
- Procedural Models
- Procedural Textures
- Procedural Animations (physics)
- Procedural Music …

Rogue, Michael Toy et al, 1980





a roguelike


Shadow Over Mordor, Monolith Prod., 2014

Minecraft, Mojang, 2009


Elite, Acornsoft, 1984

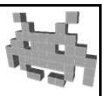

Left 4 dead, Valve, 2008


Rescue the beagles, 16x16, 2008

51

GENERAL TERMINOLOGY

# «Baking»,
# «Baked» / «Pre-baked»



*it*: "cuocere (al forno)"

once and for all, producing one asset (otherwise, it's caching)


: Storing *for good* the result of a procedural generation, for later use

often, (a refined versions of) the ones normally employed in real time

e.g.: a "baked light-map", a "baked animation"…

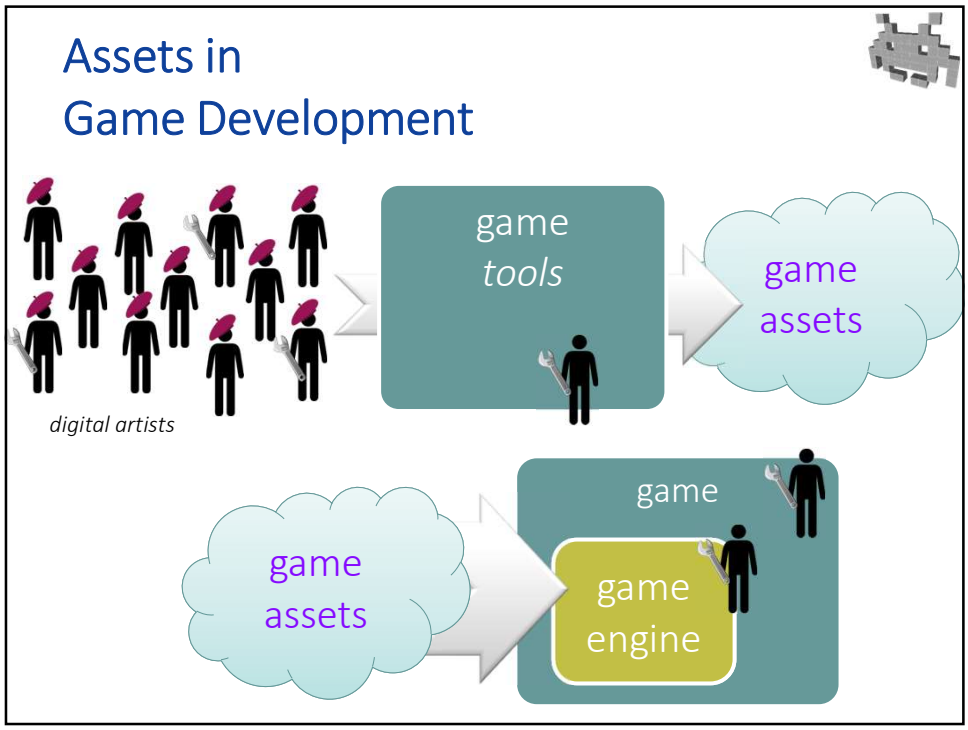
**We gain:**

- time (CPU / GPU workload)
- almost total independence from computation complexity ! → less compromises, more quality

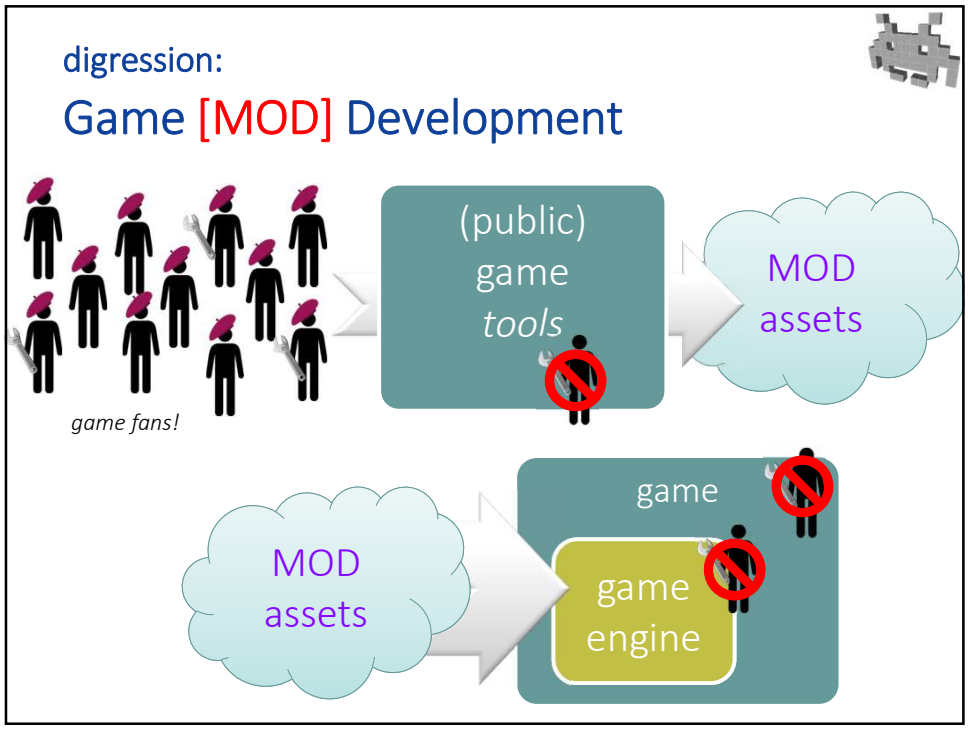
**We pay with:**

- space (on disk , Ram , GPU RAM)
- loss of flexibility (all the parameters used by the computation are frozen)

52


Assets in
Game Development
digital artists
game
tools
game
assets
game
assets
game
game
engine

53


digression:
Game [MOD] Development
game fans!
(public)
game
tools
MOD
assets
MOD
assets
game
game
engine

54

GENERAL TERMINOLOGY

## How «hard-wired» (or «hard-coded») is a given-video game feature?

- Where is it implemented?

| in HARDWARE (eg. on the GPU) | in the GAME ENGINE | in the code of that video-game | in a script (or similarly controllably by an asset) |
|---|---|---|---|

more «Hard-Wired» ⟵ ⟶ less «Hard-Wired»

- Who can modify it?

| The Hardware vendor (> platform dependence!) | the Game Engine dev-team | the tech part of the video-game dev-team | the artists (e.g. level designers); the modders |
|---|---|---|---|

55

GENERAL TERMINOLOGY

## How «hard-wired» / «hard-coded» is a given-video game feature?

**More Hard-wired**

- 👍 more efficiency
- 👍 more scalability
- 👍 more reuse

**Less hard-wired**

- 👍 ease of maintenance
- 👍 more customizability
- 👍 more flexibility

56