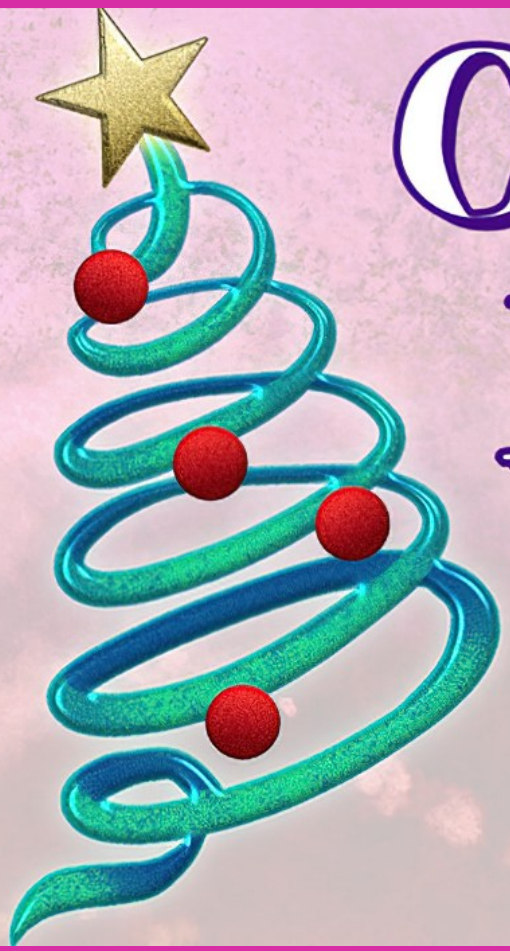


Oulu Gamedev Xmas Special

Prototyping and Core Playable

Finding the Fun

TIGER TEAM PRODUCTIONS

Ralf C. Adam

This lecture was held in November 2022 at the IGDA Game Dev Xmas Special in Oulu/Finland. The presentation may not include all original materials shown during this talk (pictures, videos etc.) –but has been enhanced with additional explanatory text which was given during the presentation in form of voice over by the speakers.

Summary &
Best Practices

Terminology

Core Playable = ∑ (All Prototypes)

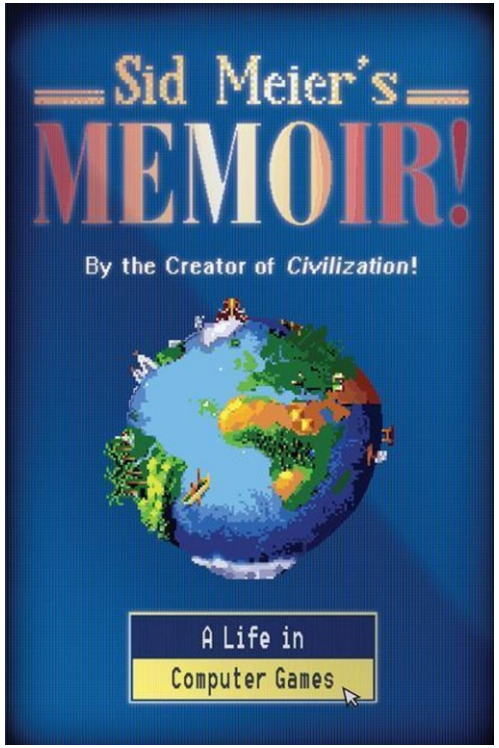
Core Playable ≠ Vertical Slice

(and hell yes! ≠ Alpha)

What is a Core Playable Build?

The sum of all Prototypes…

- Few core game mechanics
- No complete level
- No complete slice of the game in final quality
- Often consists partly or completely of placeholder graphics and assets only
- Proof of Fun:
  - Is the basic game idea fun (for hours, days, weeks)?
  - What does the player spend most of his time doing in the game?
  - “What is your game’s centre of gravity?” (Sid Meier)



Vertical Slice Version

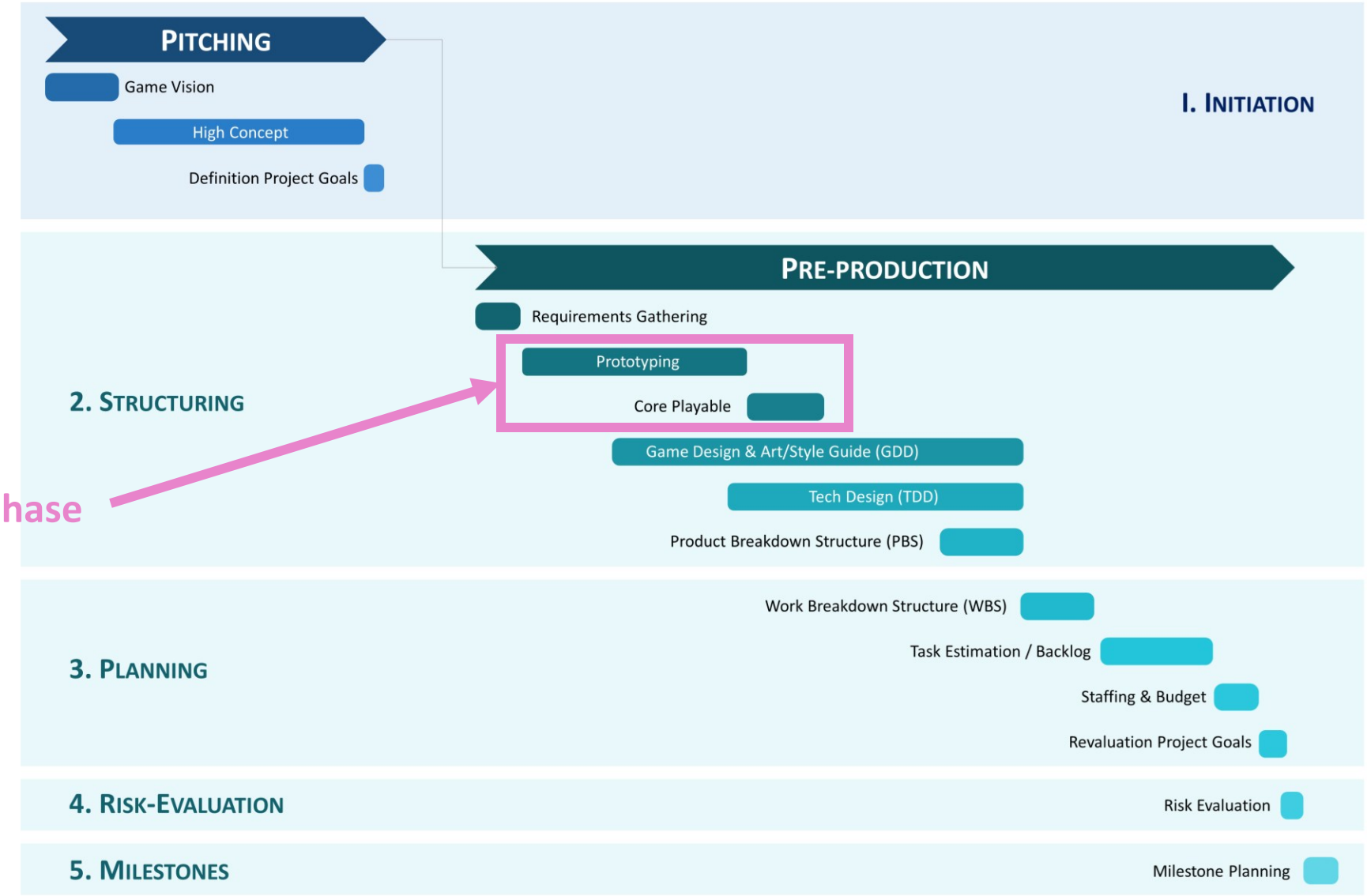
- Section of the game in final quality
- All assets in VS are in “shippable quality”
- No “throwaway code“
- Presentable to the outside world (press, trade fairs, end users/players)
- Built with final tools, pipelines and frameworks
- Final game = Vertical Slice, only more of it

What is a Core Playable Build?

Other commonly used terms

- First Playable / First Playable Prototype (FP / FPP)
- Proof of Concept Build
- Proof of Fun
- Core Prototype
- Final Prototype
- First Production Build

When is a Core Playable Build?



Alignment in team and with publisher

- Define the clear goals for your core playable build
- Provide clear contract definitions for:
  - Prototype(s)
  - Core Playable
  - Vertical Slice
  - Alpha, Beta etc.

Prototyping – Best Practice

Only most experienced team members for prototyping phase

- The smaller the prototyping team, the better:
  - It's all about speed
  - Quick agreements, iterative, "unbureaucratic
- Use the most senior team members:
  - Everyone should be able to work independently and on their own responsibility
  - A well-rehearsed approach is key!
  - No room for egos (there never is…)

Prototyping – Best Practice

One problem ➔ one prototype

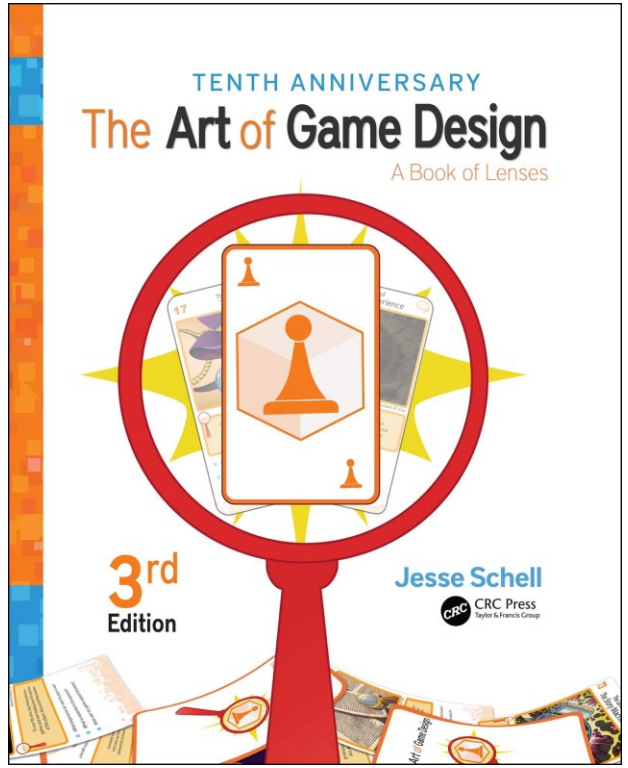
- Each prototype should answer only one question
- This question must be clearly defined and communicated upfront
- Everyone in the team must understand the goal of the prototype
- Also define the criteria for success:
  - When will you know that the prototype has fulfilled its purpose?

Prioritise & parallelise

- If possible, work on several prototypes in parallel
- Keep an eye on possible dependencies
  - Developing certain features may only make sense once others have already been implemented
- Prioritise the individual prototypes
- Never prototype already proven game mechanics
  - What for? What would you prove?

What could possibly go wrong?

- Change your perspective (also check out Jesse Schell, see book on the right).
- Instead of being confident, ask: what can go wrong?
  - Which is the default publisher perspective anyway
- Be the devil's advocate:
  - What factors might prevent the game from being fun?
- This approach will help to set the right goals and priorities for your prototypes



Prototyping – Best Practice

Don‘t waste any time!

- Efficiency beats **everything** during prototyping
- Active time boxing:
  - No prototype longer than 1 week
  - Even better: 3 prototypes a day (!)
- Remember: whatever you build, you’ll throw it away afterwards anyway - this is not about a beauty prize
- It's better to do as many iterations as quickly as possible – you'll learn a lot more
- Otherwise you run the risk of feature creep (watering down the idea and the core of your original prototype).

Prototyping – Best Practice

Fail successfully

- Prototyping = Trial & Error
- Now is the time to try out even the supposedly crazy
- It's unlikely that even one idea will work the way you imagined it right away
- Even if you only know what doesn't work in the end: it’s better to realise now that an idea isn’t so great after all than halfway through production
- Fail early, fail often, learn from it and improve your prototypes constantly & consistently

Avoid feature creep

- The problem with prototyping: you constantly have new (good) ideas.
- So keep an eye on your game vision:
  - Does a new feature support your vision? Does it make the game better? Or does it rather distract from the core game idea?
- Prototyping is about finding out what your core game idea (and therefore the fun of the game) is all about
  - Unnecessary ballast is more of a burden
- And if you inflate the scope of your prototype, how do you know what the problem is if it ends up not being fun?
- Focus, and give the Core Game Loop the opportunity to shine and convince in its sheer simplicity.

„Toy“ & „Juiciness“

- Definitions of "toy" are often different, depending on the publisher/studio you are talking to
- Often: "Core game mechanics minus any rules or decisions"
- A toy is already fun for the sake of fun, without any additional rules or mechanics
- A game, on the other hand, has a more holistic approach with defined rules and goals
- This fun with the "toy" often requires what is called "juiciness".
- *„Juicy“ = Feels alive. Every user input elicits an immediate – and satisfying – response from the environment*
- Combination of graphics, animations, sound/FX and perfect interaction between camera and controls

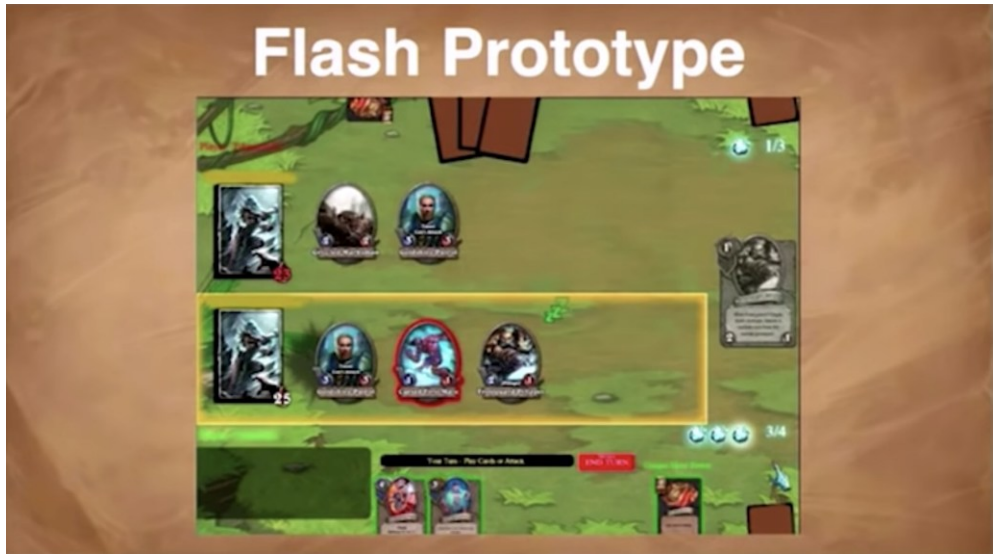
The best tool

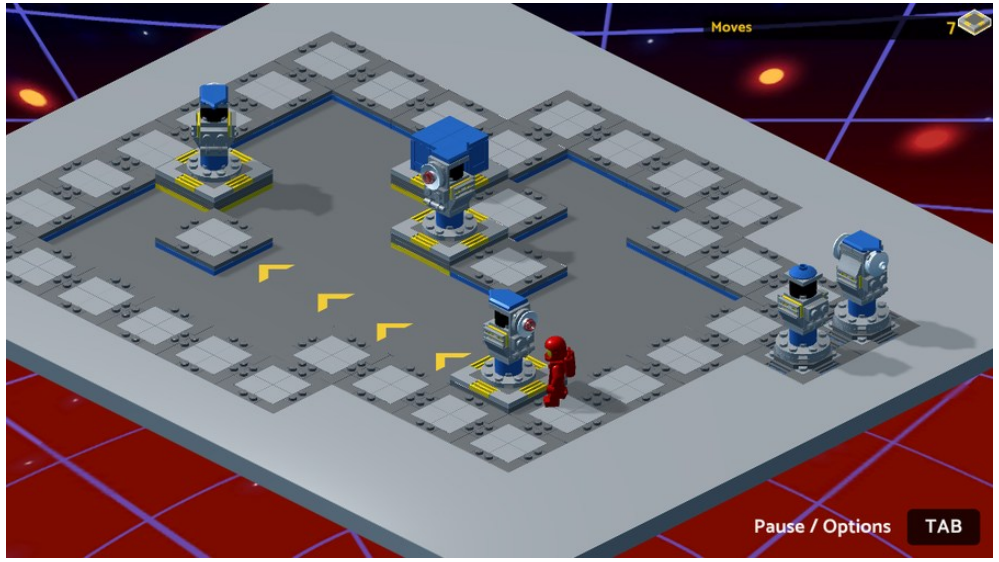
- For rapid prototyping, any tool is allowed - the faster, the better.
- At this point, the choice of engine doesn't matter (even if you want to use Unity now and Unreal later - or vice versa).
- Take what you can get - and help yourself where you can, in the asset store (helpful especially for Toy & Juiciness, see prev. slide)
- Other possibilities:
  - Gamemaker, Marmalade, etc.
  - Balsamiq, Flash
  - MS PowerPoint
  - Videos mit Adobe Premiere
  - usw.

Pen & paper prototypes

- The archetype of game design, so to speak
- Can be applied to many genres (admittedly, e.g. an RPG is better suited than a shooter)
- Extendable into other forms of "physical" game design (e.g. level design with Lego bricks etc.)
- Example: Hearthstone

  (*https://hearthstone.gamepedia.com/Design_and_development_of_Hearthstone*)





And in the end... throw it all away!

- Prototyping = Trial & Error
- Never build a final code base on a quick & dirty prototyping framework
- Even if it may sound strange: it is quicker to set up and programme everything (cleanly) a second time
- Take all the "learnings" with you and start completely from scratch
- Be prepared: might become a bit difficult at times to explain to your publisher ("What, you want to throw it all away? But we paid for it...????") -> but stick to it!

More tips and tricks

- For almost (!) every genre an elementary part of prototyping -> CCC
  - **C**amera
  - **C**haracter
  - **C**ontrols
- Balancing: always increases/decreases values in large steps
  - e.g. 100% or even 200% steps:
    - Too many baby steps make it difficult for you to see any noticeable difference at all
    - If 100% was too much, try half (i.e. 50%) and so on
    - This way you will get closer to the right value much faster
    - Often 100% is still too little - you will be surprised

Prototyping – Best Practice

Producer‘s Checklist

- ✓ Agree on the objective for core playable internally & with your publisher
- ✓ Define clear goals for each single prototype
- ✓ Tackle only one problem per prototype
- ✓ Create many smaller prototypes in parallel, rather than one large one
- ✓ Prioritize the order in which you create your prototypes
- ✓ Determine metrics on how to measure whether you were successful
- ✓ Don’t prototype already proven game mechanics
- ✓ Approach your prototypes from a "what can go wrong" perspective
- ✓ Be efficient: always try to test what you have within the shortest possible time frame
- ✓ Embrace failure: fail fast, fail a lot, learn and improve
- ✓ Avoid feature creep
- ✓ Try to make your prototypes ***juicy*** and focus on having a ***toy***
- ✓ Choose the fastest rapid prototyping tool for your purpose
- ✓ Throw away your code at the end of prototyping and start from scratch once in production

Contact

Ralf C. Adam

ralf@tigerteam-productions.de
Skype: ralfcadam
www.tigerteam-productions.de