# A concurrent component-based entity architecture for game development

Ferdinand Majerech
Supervisor: RNDr. Jozef Jirásek, PhD.
Consultant: Mgr. Samuel Kupka

Univerzita Pavla Jozefa Šafárika v Košiciach
UPJŠ

November 21, 2013

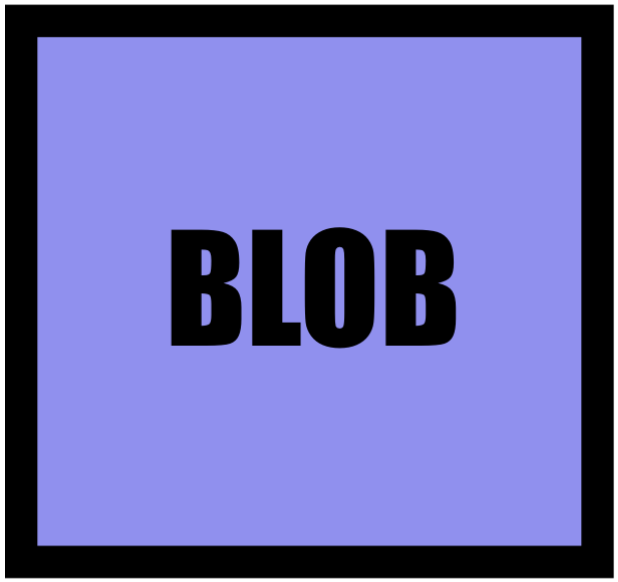
# Entities

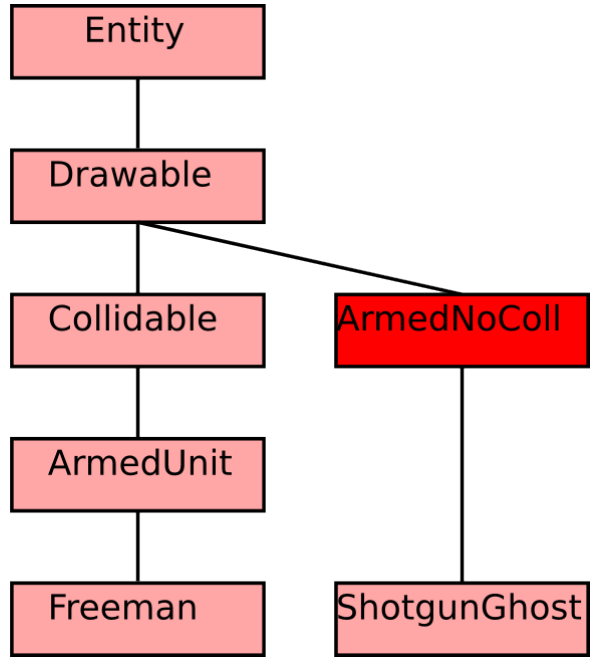
- Game entities:
  - Tree
  - Localised sound effect
  - Tank
  - Tank with a jetpack and shark launcher

# Entities - requirements

- As lightweight as possible
- Data-defined
  - Obsolete the programmer
- Easy to modify

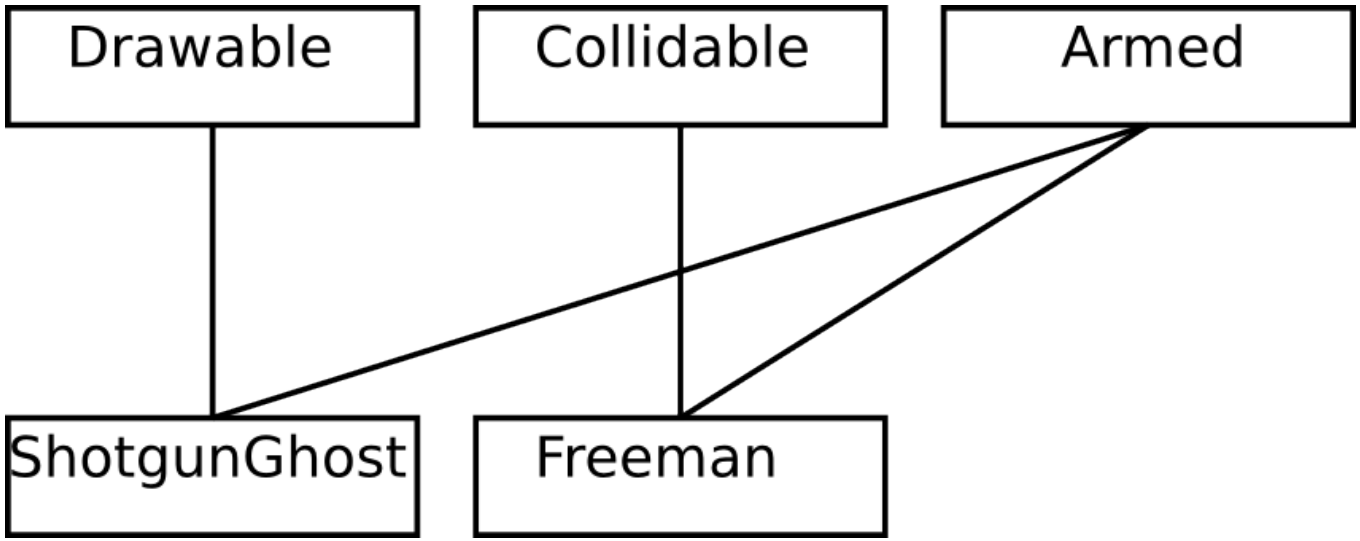
# OOP entities v1




- This is simply not sane
- Blobs can be data-defined, though

# OOP entities v2




- Virtuals everywhere
- Entities need to *implement* interfaces
  - Which will lead to *some* duplication in the *best* case
  - And can be inconsistent

# Components v1

- Entity is an ID associated with components
- Components consist of data and logic
- Components may depend on other components
- Expensive inter-component communication

# RDBMS inspired components (current)

- Components are plain data (columns)
- 1 or 0 components of any type per entity
- Logic is separated into *systems*
  - A system specifies component types it processes.
  - Entity system calls the process for every entity with matching components.
- Entities fully defined in data, emergent behavior
- In some frameworks entities can be modified at run time
- Systems affect each other during an update
- Threading is still difficult

# Tharsis (goals)

- Entity system designed for threading
  - *Processes* (systems) don't affect each other during an update
  - Processes automatically assigned to threads at run-time (unless overridden)
  - Avoid locking where possible
- Lightweight
  - Avoid cache misses and virtuals, inline what we can
  - Process-specific generated code should minimize execution overhead
- Easy to use and type-safe (with compile-time validation)
- Data-driven entities that can be efficiently altered at run time
- Builtin statistics; profiling?

# A typical multi-threaded game (X:Rebirth)

- Main Thread 1
- Main Thread 2
- Graphic Driver Code Thread
- Path Finding Thread
- Sound Thread
- Loading Thread
- Workers where spawning won't murder performance

# A (hypothetical) Tharsis game

- Any new piece of functionality is a new process.
- Physics process
- Locomotion process
- Visual process
- Controller process
- Script process
- Collision process
- Spawner process
- Sound process
- Health process
- Weapon process
- ...
- If enough cores, a process will be in a separate thread
- Otherwise multiple processes will share a thread

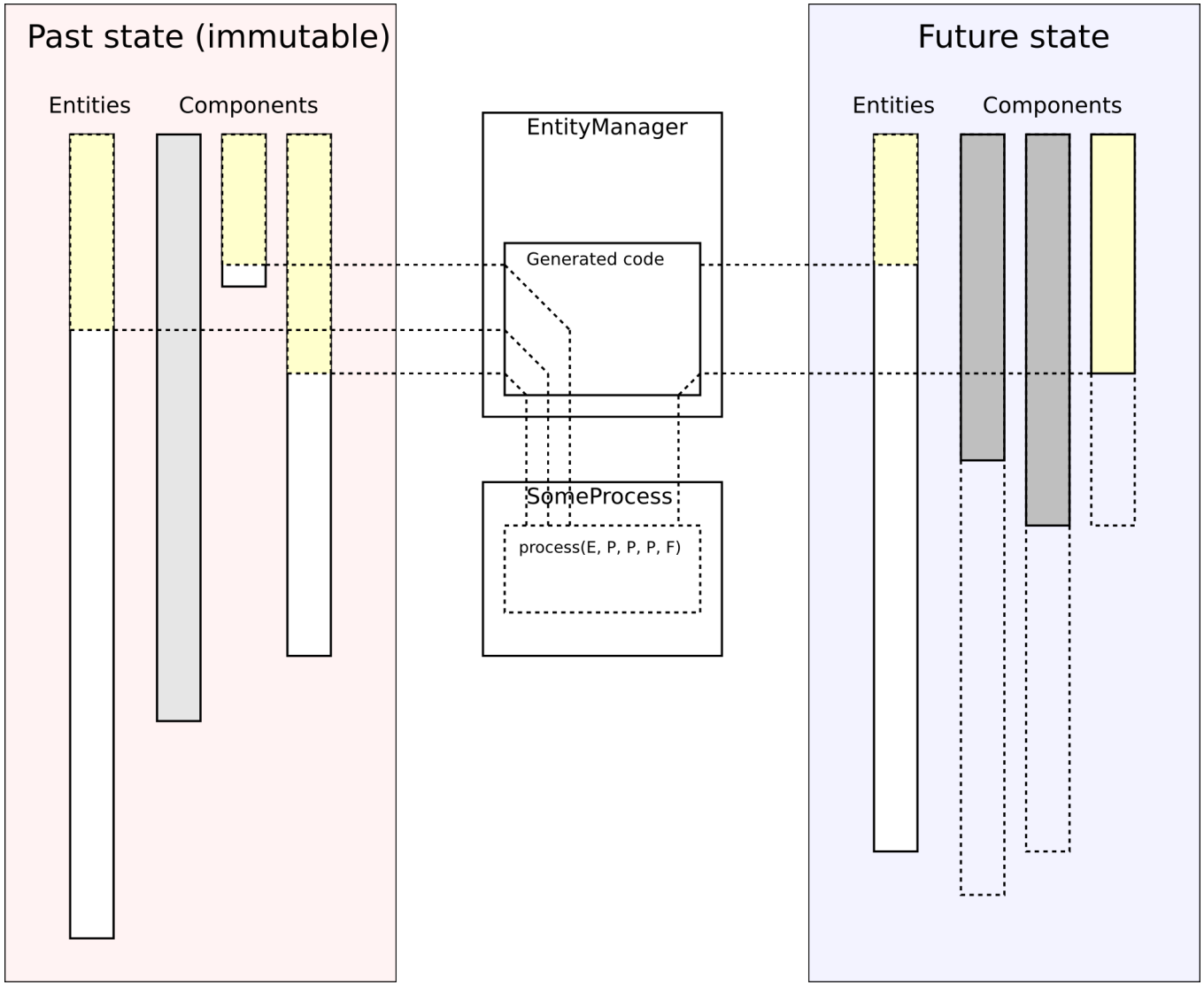
# Past and future

- Immutable data can be read without locking
- *Past*: a copy of all game state from the previous update
  - Processes don't affect each other during an update
  - Processes can read data without locking
- *Future*: game state generated by processes
  - Generated on the fly, leaving no gaps
  - To remove a component, don't copy it to future
  - A process can only write one component type

# Memory organization

- Arrays (past, future entities)
- More arrays (past, future components of all types)
- Even more arrays (auxiliary data (TBD))
- CPUs like arrays a lot
- Preallocate everything, only allocate in emergency

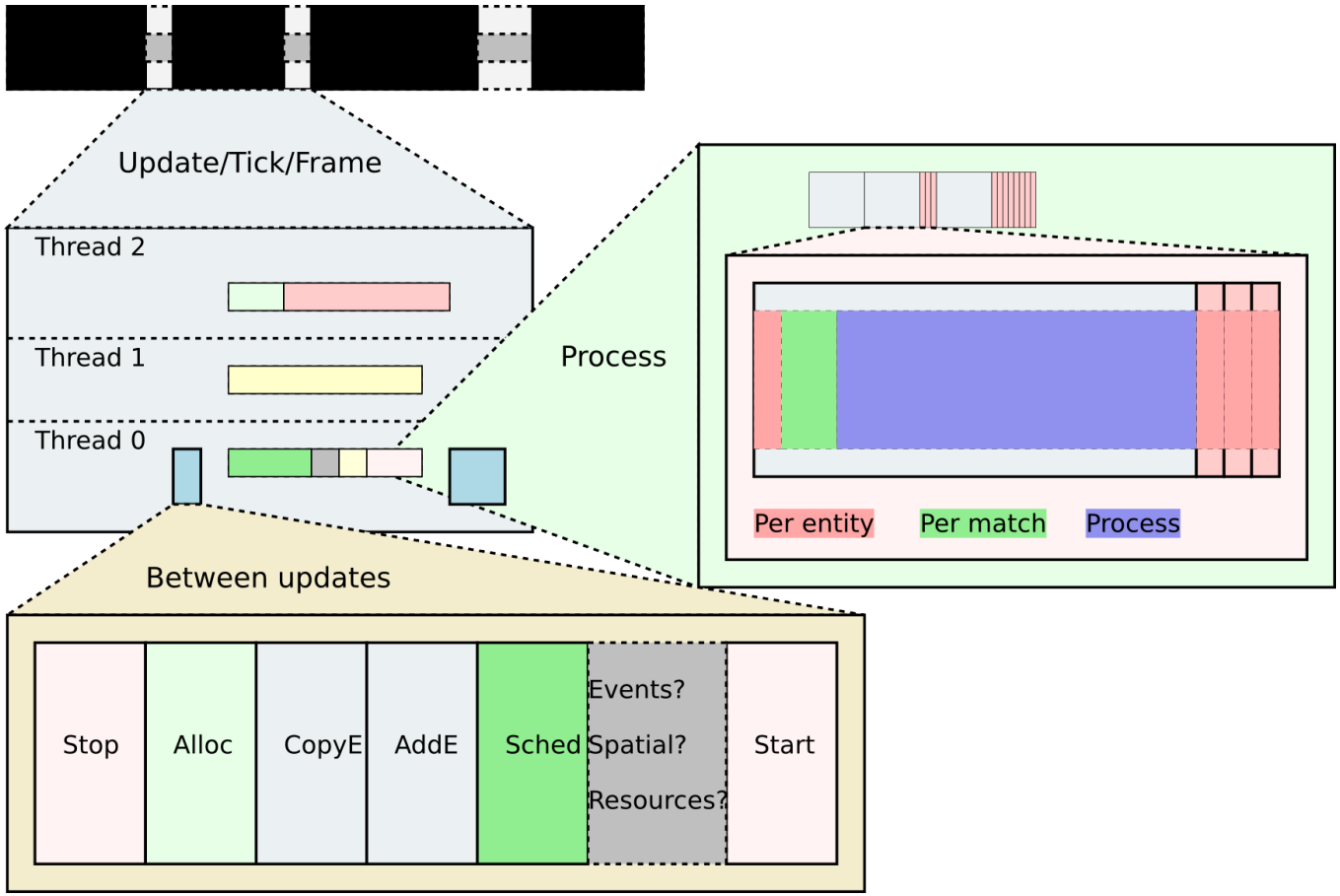
# Memory organization



# Overhead

- Between updates
- Per entity
- Data must still be copied if a process is not scheduled to run

# Overhead


Update/Tick/Frame
Thread 2
Thread 1
Thread 0
Process
Per entity
Per match
Process
Between updates
Stop
Alloc
CopyE
AddE
Sched
Events?
Spatial?
Resources?
Start

# Scheduling

- TBD
- Ensure equal load, avoid long updates
- Processes should be able to skip updates if needed

# Uncertainties

- Inter-system communication
- Spatial management
- Resource locking (entity creation, resource loading)

# Sources


- Main inspiration: Adam Martin. *Entity Systems are the future of MMOG development* (2007)
- Chris Stoy. *Game Object Component System* Game Programming Gems 6 (2006)
- Michael A. Carr-Robb-John. *The Game Entity* Game Developer Magazine November 2011
- Terrance Cohen. *A Dynamic Component Architecture for High Performance Gameplay* GDC Canada (2010)
- Tony Albrecht. *Pitfalls of Object Oriented Programming* Game Connect: Asia Pacific 2009

# Roadmap

- Get all features to work without threads
  - Blog
- Get threads to work
  - Fix unexpected issues
  - Blog
- Scheduling
  - Blog
- Write paper
  - Blog?

# More info

- D language
- Open source, Boost license
- Platform independent (x86 atm, ARM as D support matures)
- Source: https://github.com/kiith-sa/Tharsis
- Design: https://github.com/kiith-sa/Tharsis/blob/master/tharsis.rst
- Blog (not there yet): http://defenestrate.eu