# OOP is for Boomers

Милош Суботић

20. јануар 2024.

# Case study

Chicken invasion
https://youtu.be/AtoN1hgz0Nc

# Constraints

- MVP (Minimum Viable Product)
- Логика игрице (Gameplay) је задатак, графику и звук неко други ради.

# Декомпозиција

- У оквиру логике, шта су нам под-задаци?
- Шта можемо занемарити?
- Шта ћемо делити са екипом за звук и графику?

# Task Capturing

# Декомпозиција

- Структуре података
- Алгоритам

# MVP

Занемарити:

- HUD (Head-Up Display): животи и поени
- Анимације: експлозије, recoil, махање крилима...

# Шта делимо?

- Структура података
- Баш, баш делимо или само да кажемо шта да цртају?

# Алгоритам

- Шта је ту главни алгоритам?

# Алгоритам

- Collision detection
- Сваки са сваким?

# Како у C-у

- Па како?
- Да искодујемо

# Проблеми

- `switch` је $\mathcal{O}(n)$

# Решење

- Низ показивача на функције
- Имамо `collision()`, `reaction()`, `anim()`, `draw()`
- За сваку свој низ да направимо?
- Ако додајемо нови објекат?

# Решење решења

- Структура са показивачима
- На чега вас ово подсећа?

# Структуре

- Да нацртамо нешто
- Наслеђивање неко?
- Композиција нека?

# Структуре

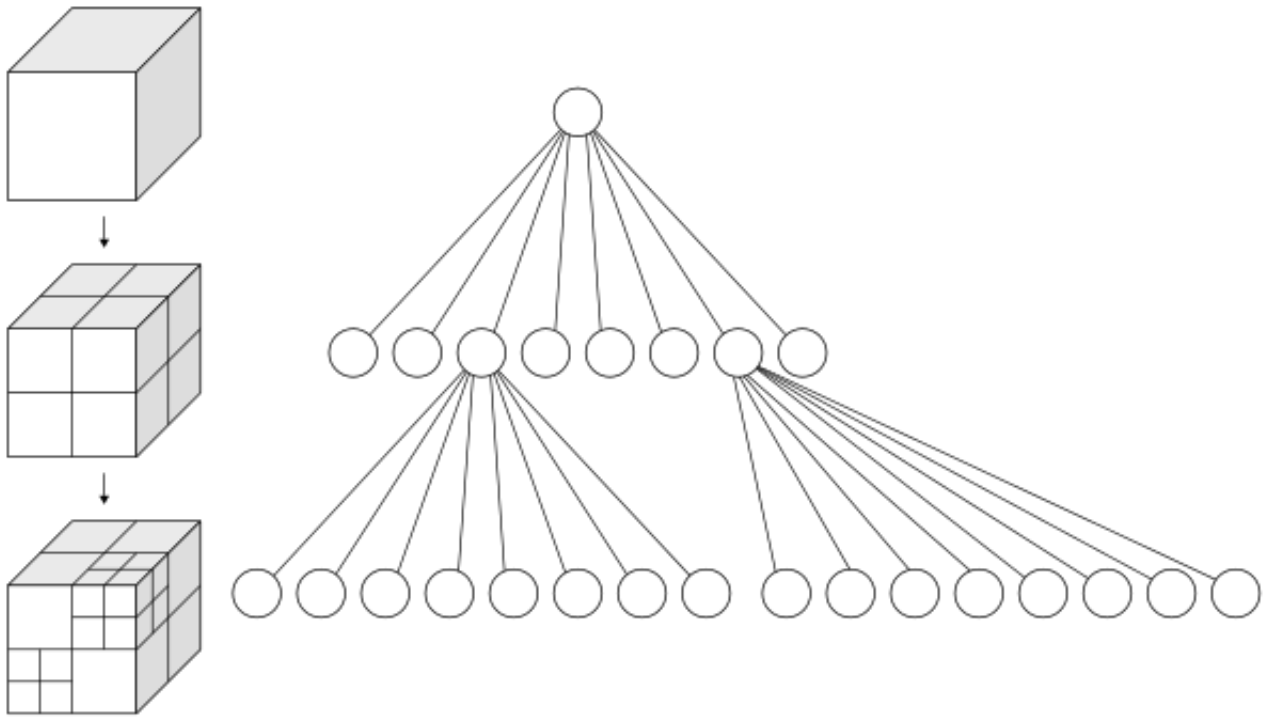
- Композиција: вектор са кокошкама, вектор са ракетама...

# Како у C++

- Па како?
- Да искодујемо

# Оптимизација

- У плану
- Октално (квадратно) стабло



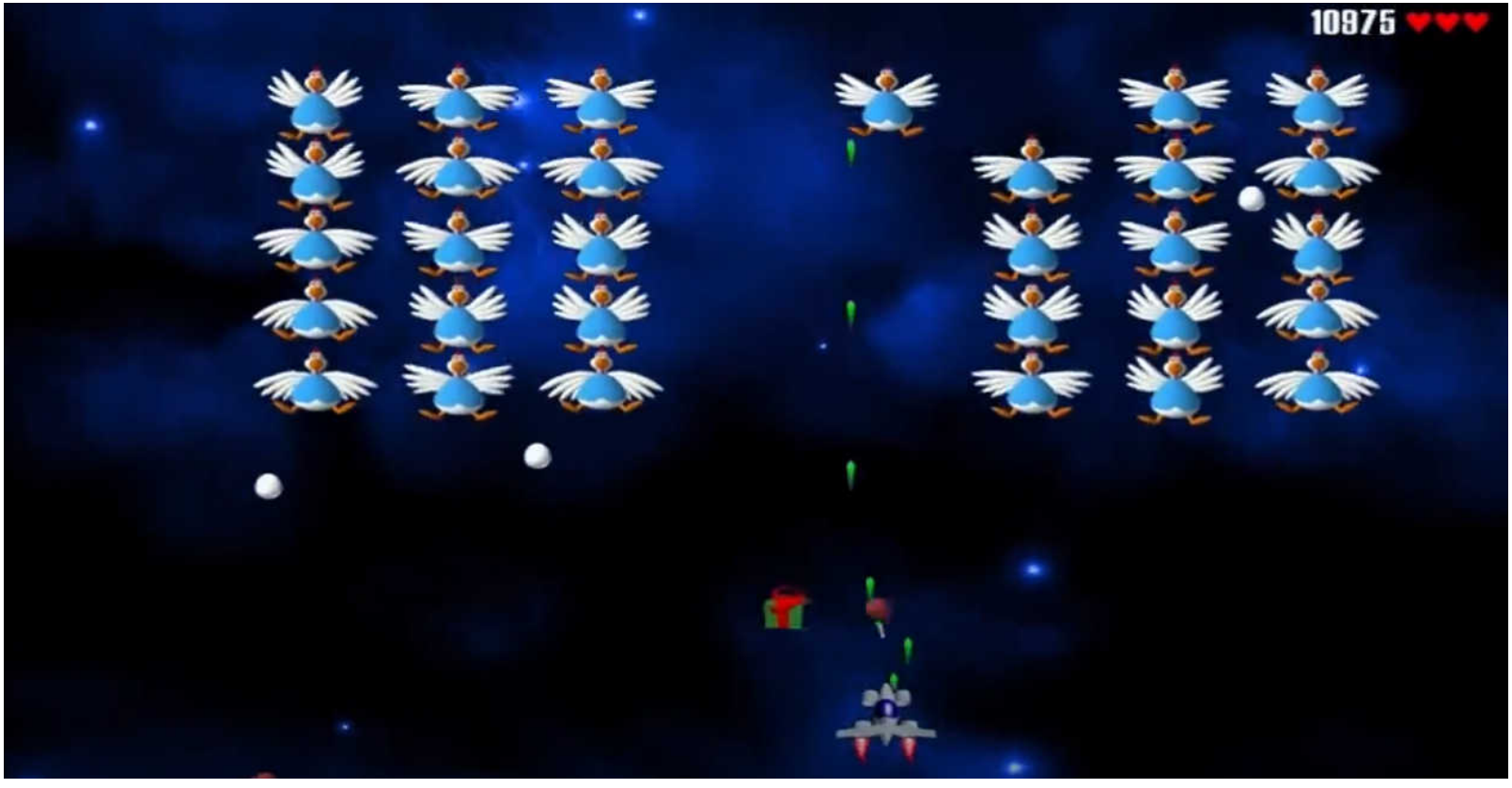
# Структуре

- Композиција: квад стабло са кокошкама, квад стабло са ракетама...?

# Рани pre-alpha

- Успех! Имамо MVP

# Компликације



# Још компликације

# Компликације компликација

# Проширивање

- Како и где додати нове ствари?
- Проблеми у колаборацији.

# Проширивање

Шта ако проширујете неки тамо пројекат? Неки open-source?

- **Убедити аутора** да је то добра идеја.
- Кад би га сви убедили **класа би била огромна**
- **Не можеш се предомислити** а да се то не обије другим корисницима о главу.

# Проширивање

Па можемо да наследимо?

- Ако је првобитни аутор остави омогућност (законом забранити `private`).
- Имаће неко глупаво име типа `MyHandler, RGB2`

# Значи

- Лепо је то да су методе везане за податке, али било би згодно да се лакше додају.

# Слободна функција

- Семантички нема смисла да је метода.
- Треба да је слободна функција између две класе.
  - Како бројеви. Овде само треба да буде оператор $\cap$.
  - Али како онда из квад стабла да прослеђујемо податке?

# Expression problem

- Проблем изражавања се проширивањем и модуларношћу статички типизираних класа.
- Како дефинисати нове класе проширене са новим подацима и новим методама **без превођења постојећег кода**.
- Показује проблеме програмерских парадигми и језика.
- И са Љетом Господњим 2023 **још није решен проблем!**

# Отпремање (Dispatch)

| Ниво отпремања | Синтакса | Отпрем. аргум. | Ред израж. | Снага израж. |
|---|---|---|---|---|
| Никакво | `f(`$x_1$`, `$x_2$`, ...)` | `{}` | $\mathcal{O}(1)$ | конст. |
| Једноструко | $x_1$`.f(`$x_2$`, ...)` | `{`$x_1$`}` | $\mathcal{O}(\|X_1\|)$ | лин. |
| Вишеструко | `f(`$x_1$`, `$x_2$`, ...)` | `{`$x_1$`, `$x_2$`, ...}` | $\mathcal{O}(\|X_1\|\|X_2\|...)$ | експ. |

# Да видимо како то ради

- Јулија
- C++

# Multiple dispatch is future

## DifferentialEquations + Measurements

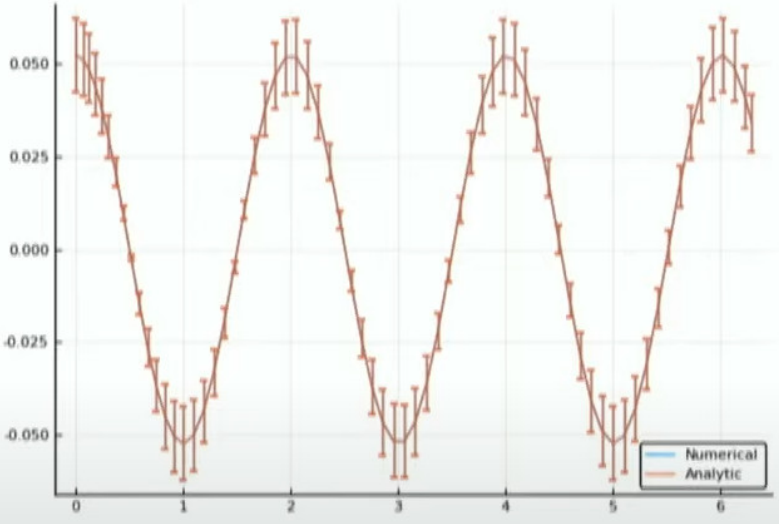
```
g = 9.79 ± 0.02 # Gravitational constants
L = 1.00 ± 0.01 # Length of the pendulum

# Initial speed & angle, time span
u₀ = [0 ± 0, π/60 ± 0.01]
tspan = (0.0, 6.3)

# Define the problem
function pendulum(du, u, p, t)
    θ  = u[1]
    dθ = u[2]
    du[1] = dθ
    du[2] = -(g/L)*θ
end

# Pass to solvers
prob = ODEProblem(pendulum, u₀, tspan)
sol = solve(prob, Tsit5(), reltol = 1e-6)

# Analytic solution
u = u₀[2] .* cos.(sqrt(g/L) .* sol.t)
```




*Rackauckas et al. DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. 2017. (Journal of Open Research Software)*

*Giordano. Uncertainty propagation with functionally correlated quantities (arXiv: 1610.08716)*

# Јулија је супер јер

Јулија има врло интересантну имплементацију:

- JIT (Just in Time)
- LLVM backend

# Јулија је супер јер

- "Прерана оптимизација је корен свих зала!"Неки деда програмер из 60-тих.
- Јулија има решење:
  - Динамичко типизирање (због Пајтону)
  - Статичко типизирање (због C++)
- + веома лако се wrap-ује C/FORTRAN

# Јулија је супер јер

Добар еко систем:

- Долази са гомилом библиотека
- Scientific computing $\subset$ Technical computing
- Machine Learning

# Две врсте code-reuse-a у Јулији

- **генерички алгоритми** коришћени са различитим типовима
- **заједнички типови** дељени између различитих пакета

# Ствари које још нису решене

- Мало лакше wrap-овање C++-а.
- Да неко реши plot-овање већ једаред.

# Хвала на пажњи