CrossyLogs

Game Engine Parallelization

Nick Bonavia, Geordie Jones, and Warren Riley

Intro

- Why study parallel computing within game design
- How we made our game engine
- What our game is

Why Study Parallelization

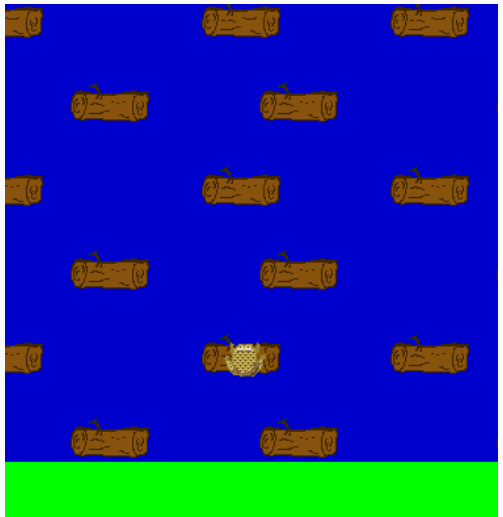
- Game consoles are adding more cores to their systems every year
- Many game applications can be parallelized
- It allows game creators to stay up to date with industry standards

Our Game Engine

- We wrote it in C++
- Utilized SDL2 to render our images
    - Also for collision detection
- Parallelized it with OpenMP

About Our Game

- Clone of Frogger 1981
- Same controls



Background and Research

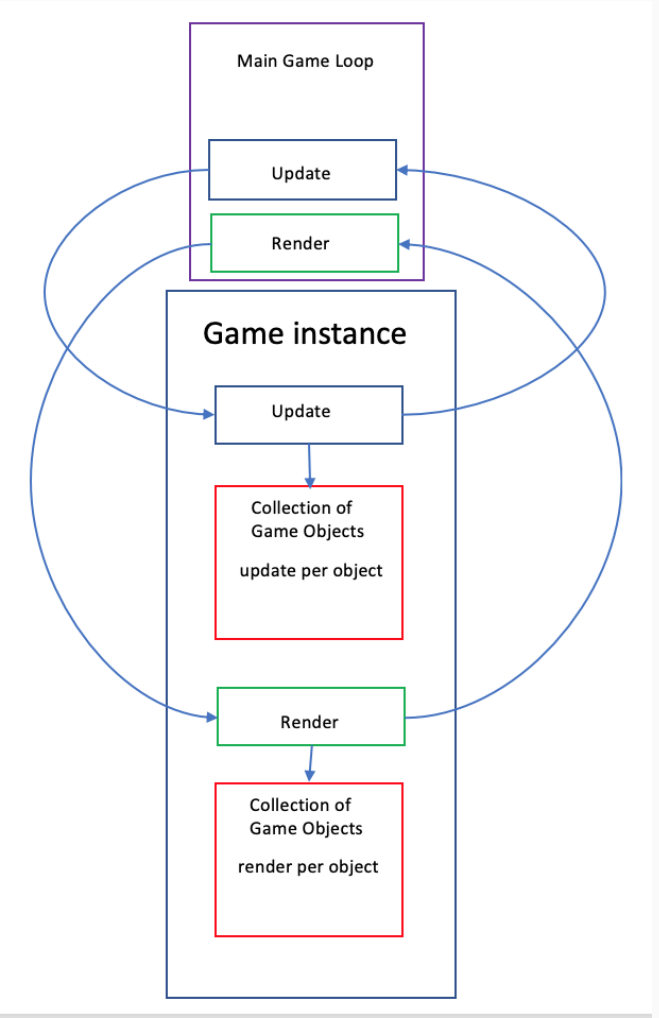
Design Patterns in Game Programming- Game Programming Patterns.

Main Game Loop- user input, update(), render()

Frame rate control measures- delta

Game Instance- collection of objects, update, render

OOP- Frog and Log are derived from Game Object



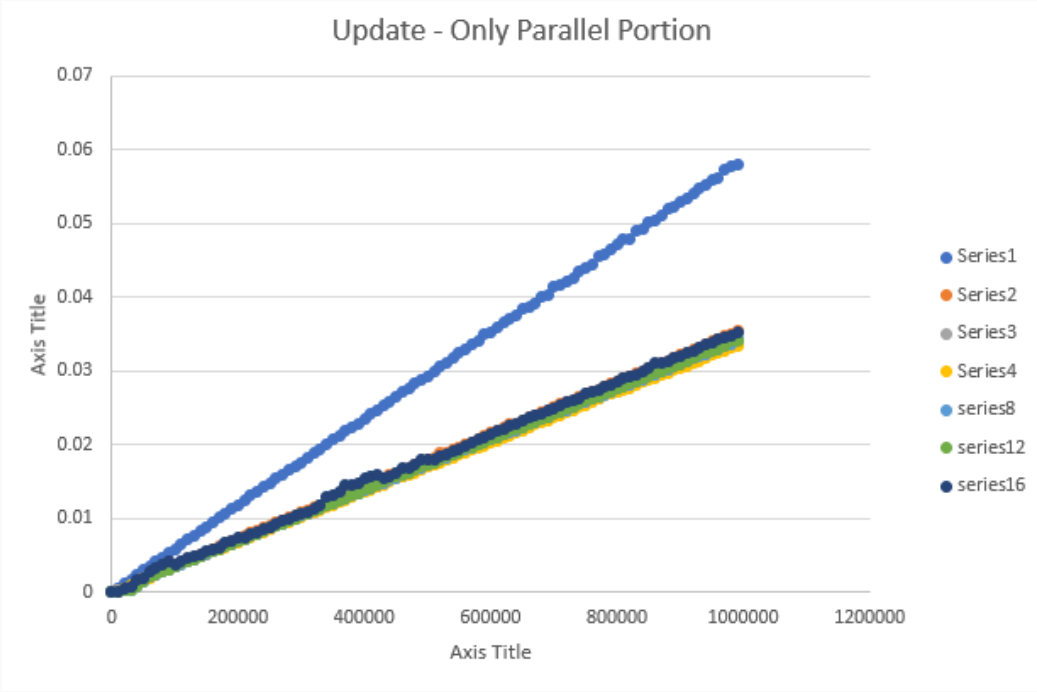
Parallelization Technique

```
#ifdef __PARALLEL__
#pragma omp parallel for num_threads(
    NUM_THREADS) collapse(2)
#endif
  for(int i =0;i<rows;i++){
    for(int j =0; j < COLS;j++){
      log_obj[i][j]->Update(delta);
    }
  }
```

- How OpenMP handles for loops
- How collapse works

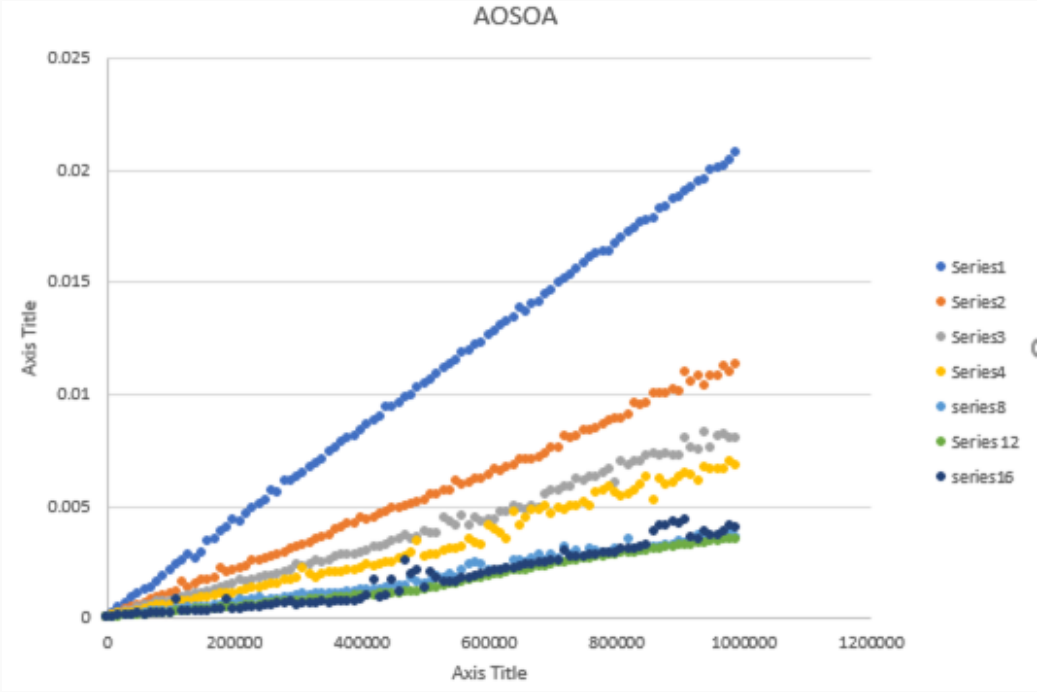
Analysis

- NumLogs = 5,940,060
- Speedup_4 = 1.7319
- Efficiency_4 = 0.4329



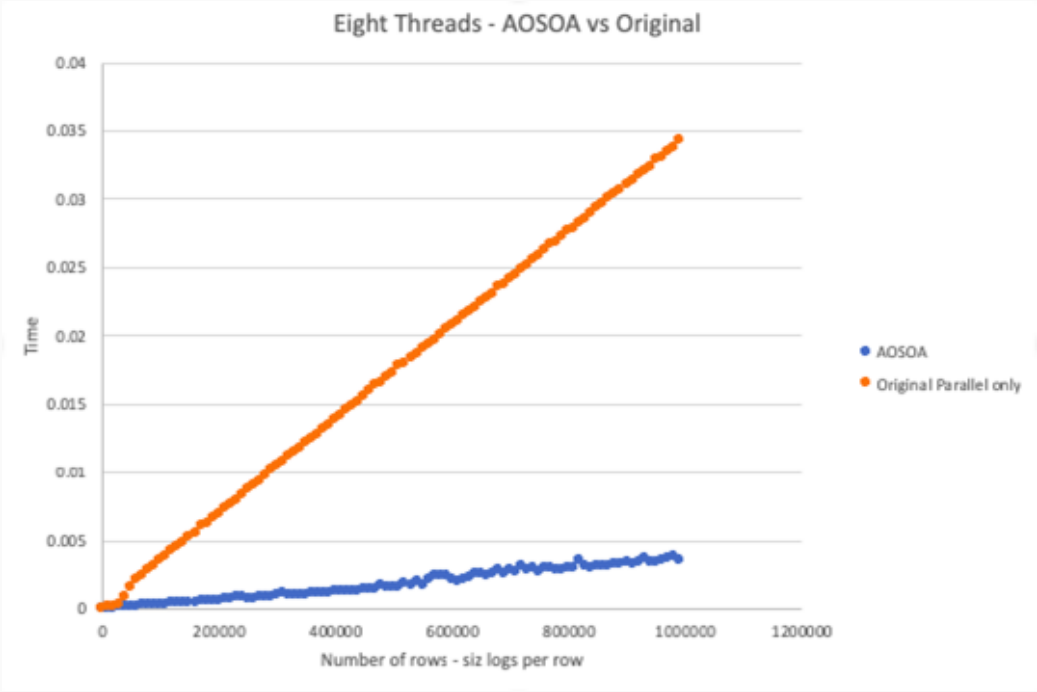
Analysis

- NumLogs = 5,940,060
- Speedup_8 = 5.7611
- Efficiency_8 = 0.72014



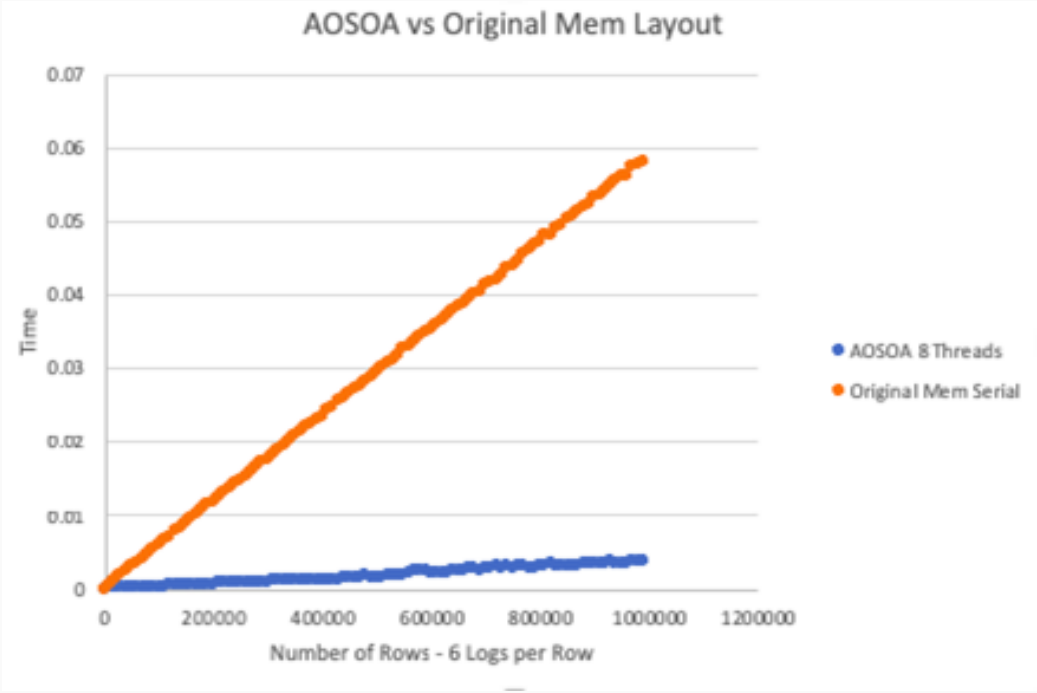
Analysis

- NumLogs = 5,940,060
- Speedup_8 = 9.5381



Analysis

- NumLogs = 5,940,060
- Speedup = 16.1387



Final point

Data Oriented design increased our parallelization speedup!