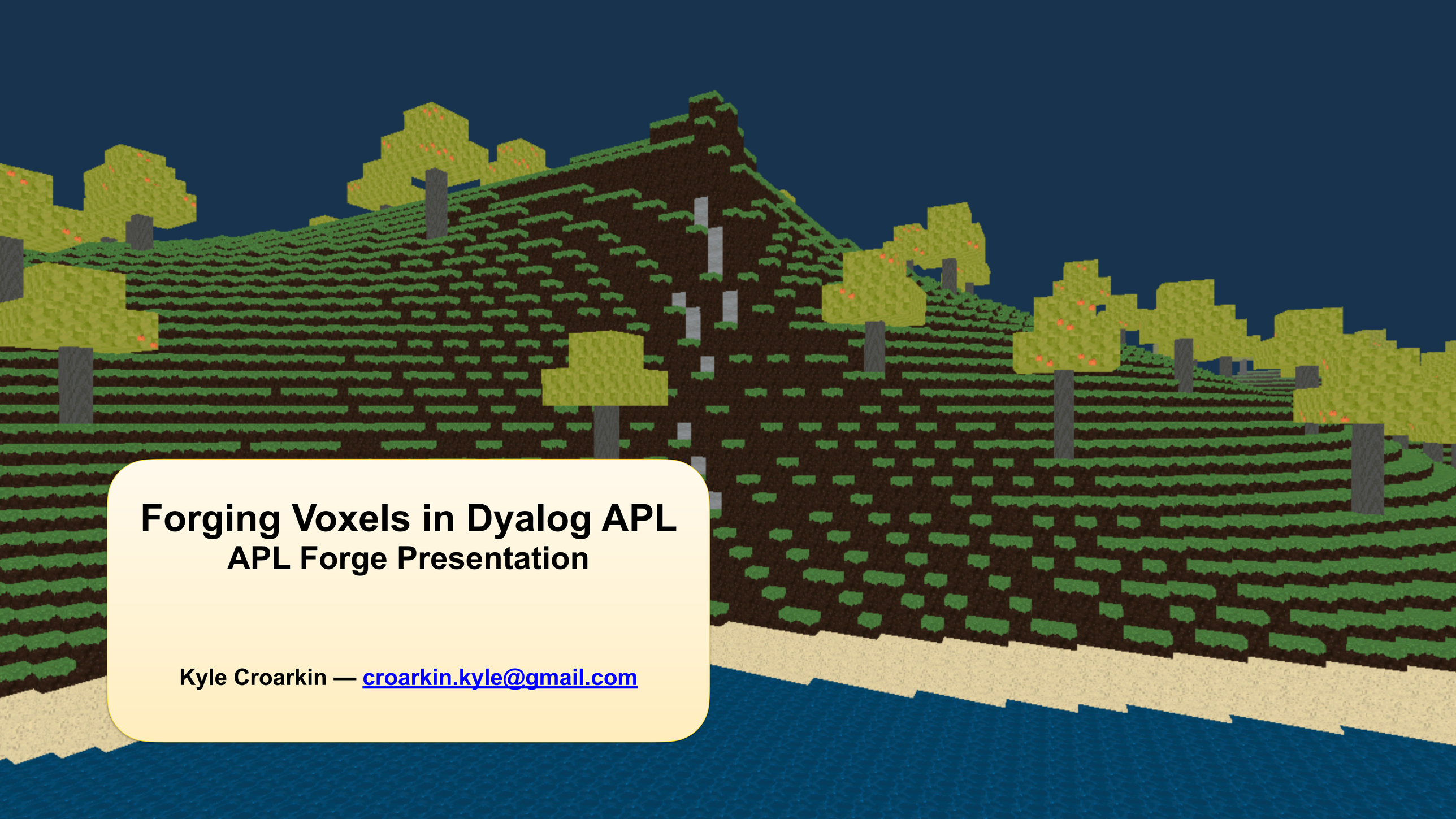


Forging Voxels in Dyalog APL

APL Forge Presentation

Kyle Croarkin — croarkin.kyle@gmail.com

- About Me
- Making a Game
- Overview
- Demo
- Architecture
- Conclusion

About me

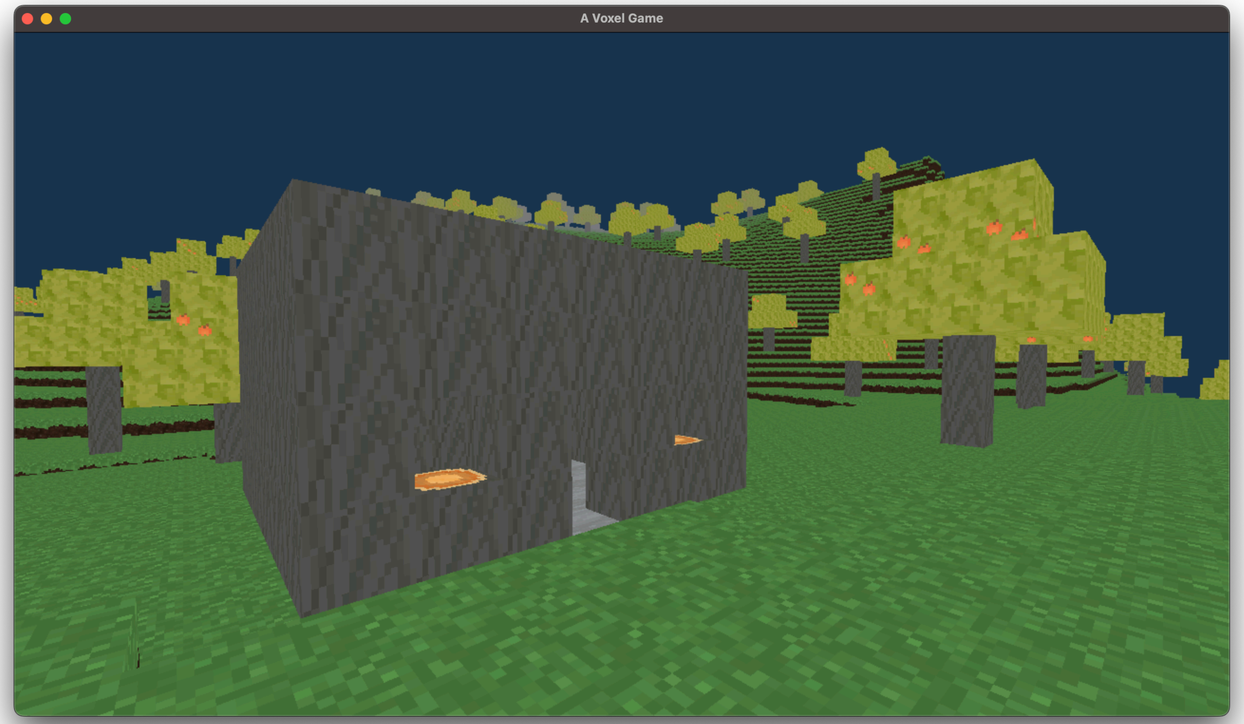
- Discovered APL 2025
- Wanted to use in known area
  - Games: bread and butter
- How does APL differ?

Requirements for Games

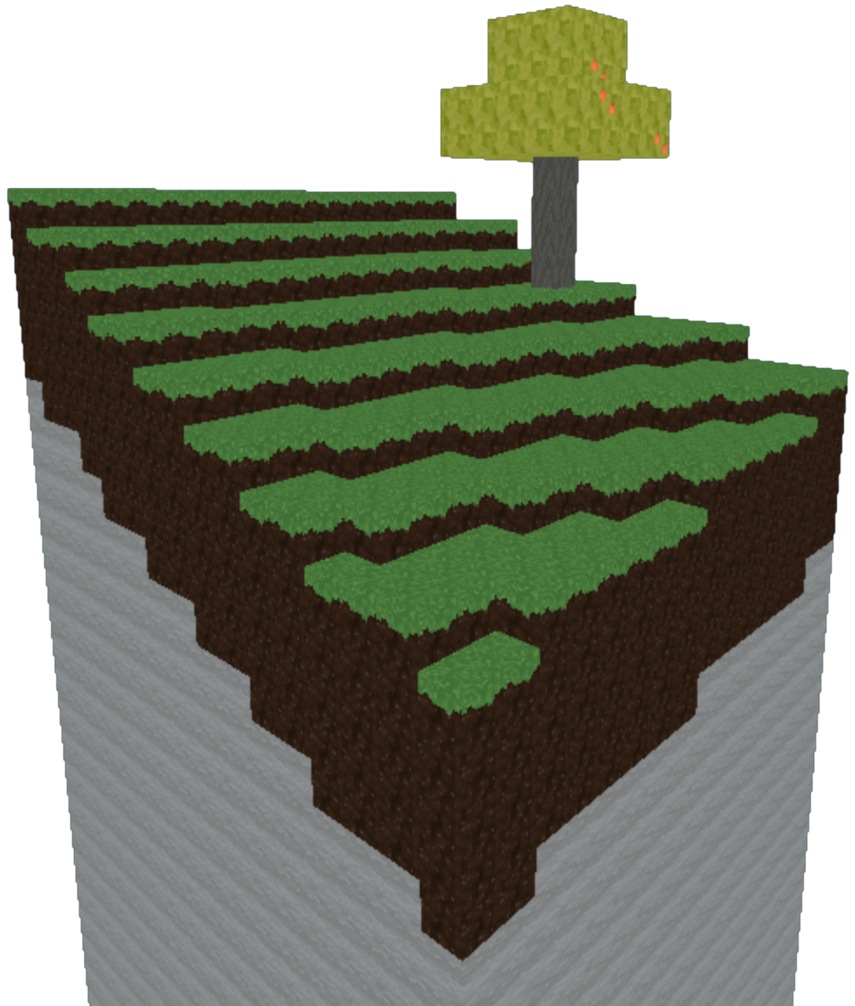
- Need good performance.
  - 60 frames per second minimum `÷60`
  - Lot of procedures need to fit in these frames
- “Big ball of state”
- Need to quickly iterate
- APL has to be a “tool of thought”
  - Goal: Use APL as much as possible

Voxel Games

- Composed of 3d blocks.
- Demo time!



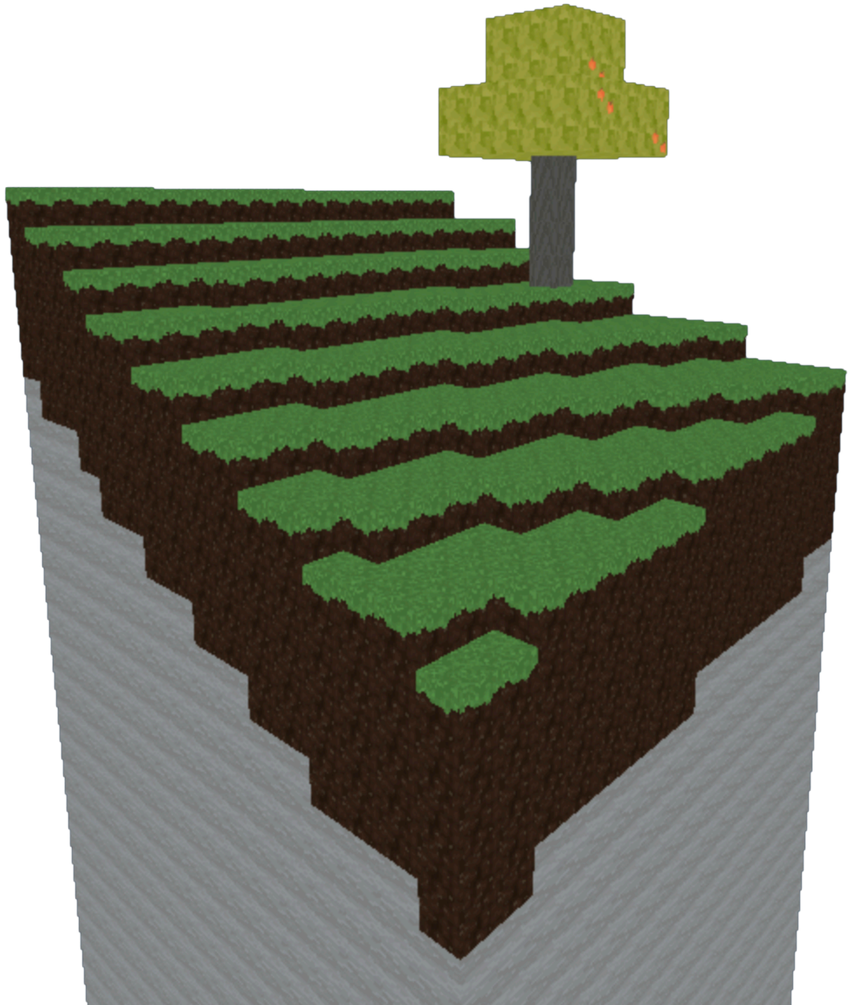
Showcase of Architecture



Chunks!

- Blocks stored as 3d arrays “slices” of the world
- Needed when the player is near them and saved to disk when not

Showcase of Architecture



What do we need to represent them?

- Their positions
- How many triangles
- Modification status
- Video memory resources

Showcase of Architecture

We can represent it with this:

- 4d array
- Inverted table

```
l ← 16 128 16

chunks ← 0ρ⍤0,l
cnames ← 'cx' 'cz' 'vb' 'wvb' 'idx_cnt' 'widx_cnt' 'dirty'
chunk_info ← ⍬⍤⍤cnames

Ci ← cnames⍳⍳⊆
Cg ← {
    ⍵≡'xz':↓⍉↑chunk_info[]⍤⊂Ci'cx' 'cz'
    ⍵≡'enc_xz':(2*16)⊥↑chunk_info[]⍤⊂Ci'cx' 'cz'
    chunk_info⊃⍤Ci ⍵
}
```

Showcase of Architecture

Adding a new chunk

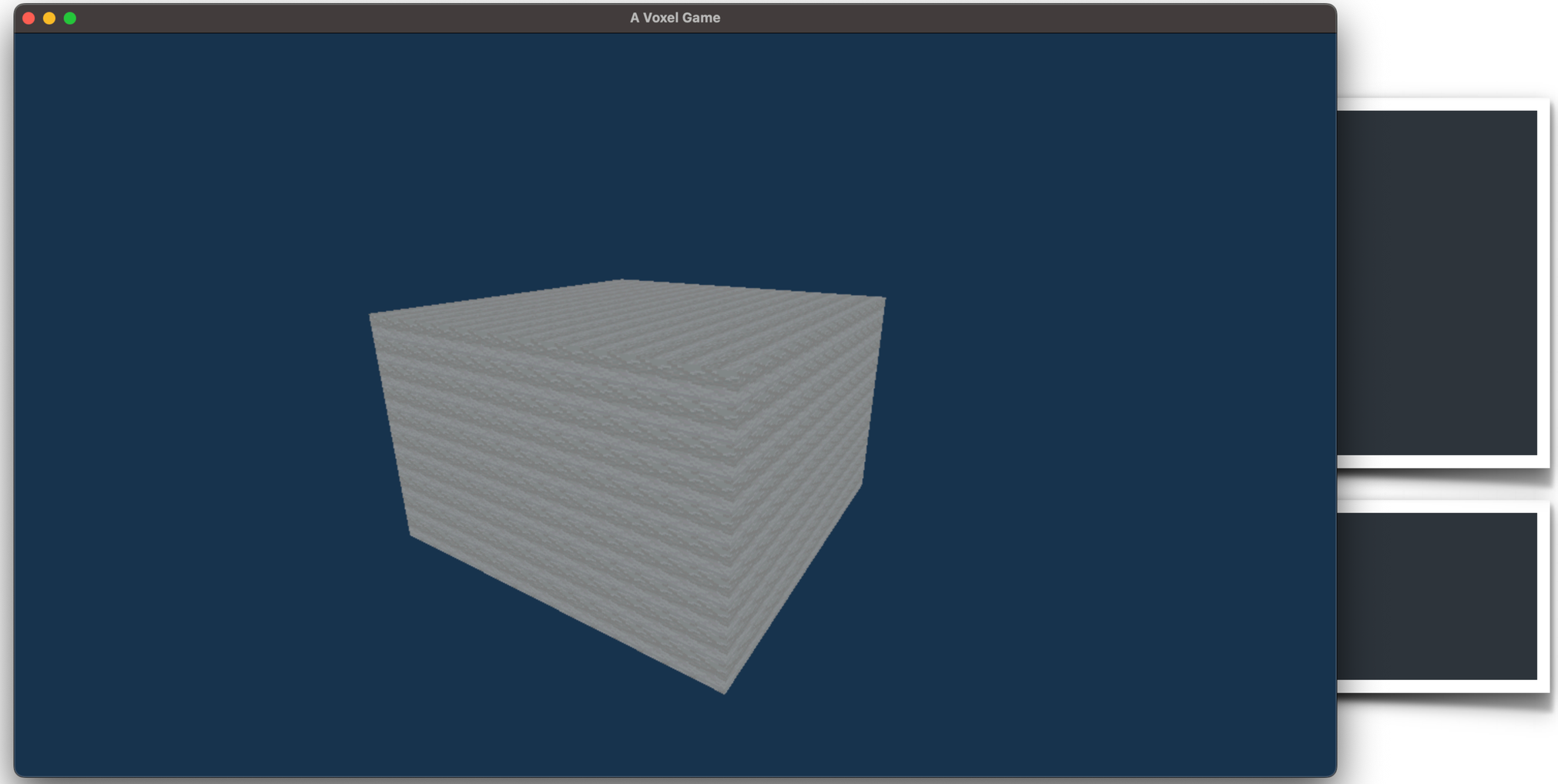
```
(LEAF DIRT GRASS SAND STONE)←(…)
blocks←lρ0
blocks[;⍳10;]←STONE
(x y)←(0 0)

chunk_info,⍤←x,z,1↑⍤2-≢cnames
chunks,←blocks
```

Unloading a chunk

```
m←~(Cg'xz')∊⊂0 0
chunks⌿⍤~←m
chunk_info←m⌿⍤chunk_info
```

Showcase of Architecture

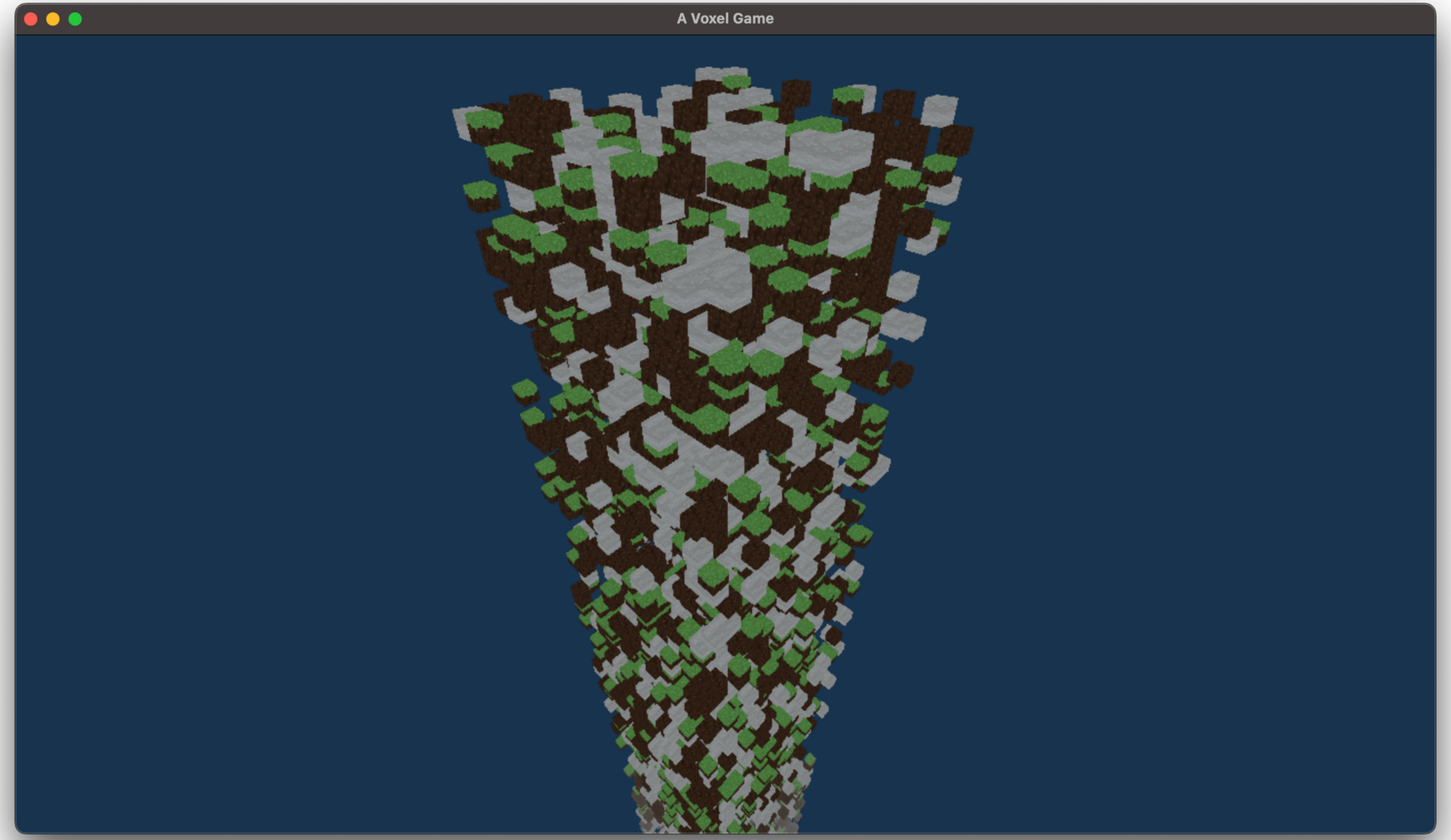


Let’s try something else

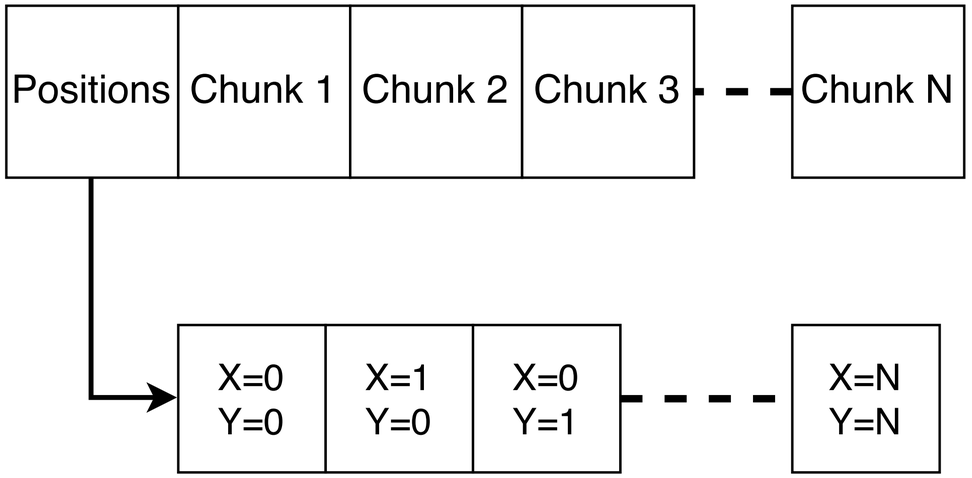
```
(LEAF DIRT GRASS SAND STONE)←(…)
blocks←⍳⍴0
blocks←(GRASS DIRT STONE,12⍴0)[?⍳⍴15]
(x y)←(0 0)

chunk_info,¨←x,z,1↑⍨¨2-≢cnames
chunks,←blocks
```

Let’s try something else



World Saving

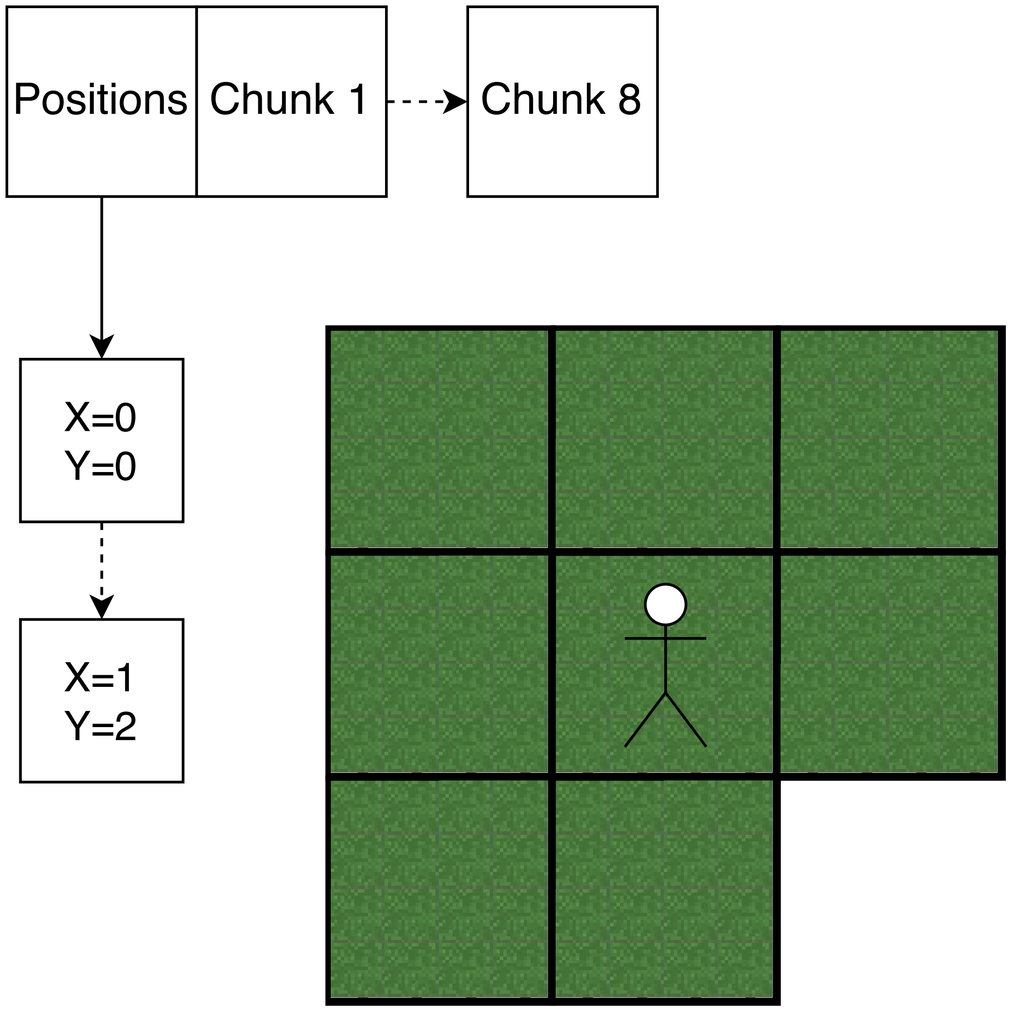


Component File

APL Component files and hashed arrays (1500⌶)!

```
map_file←save_dir,'/map_data.dcf'
map←map_file ⎕FSTIE 0
mapi←(1500⌶)⎕FREAD map 1
```

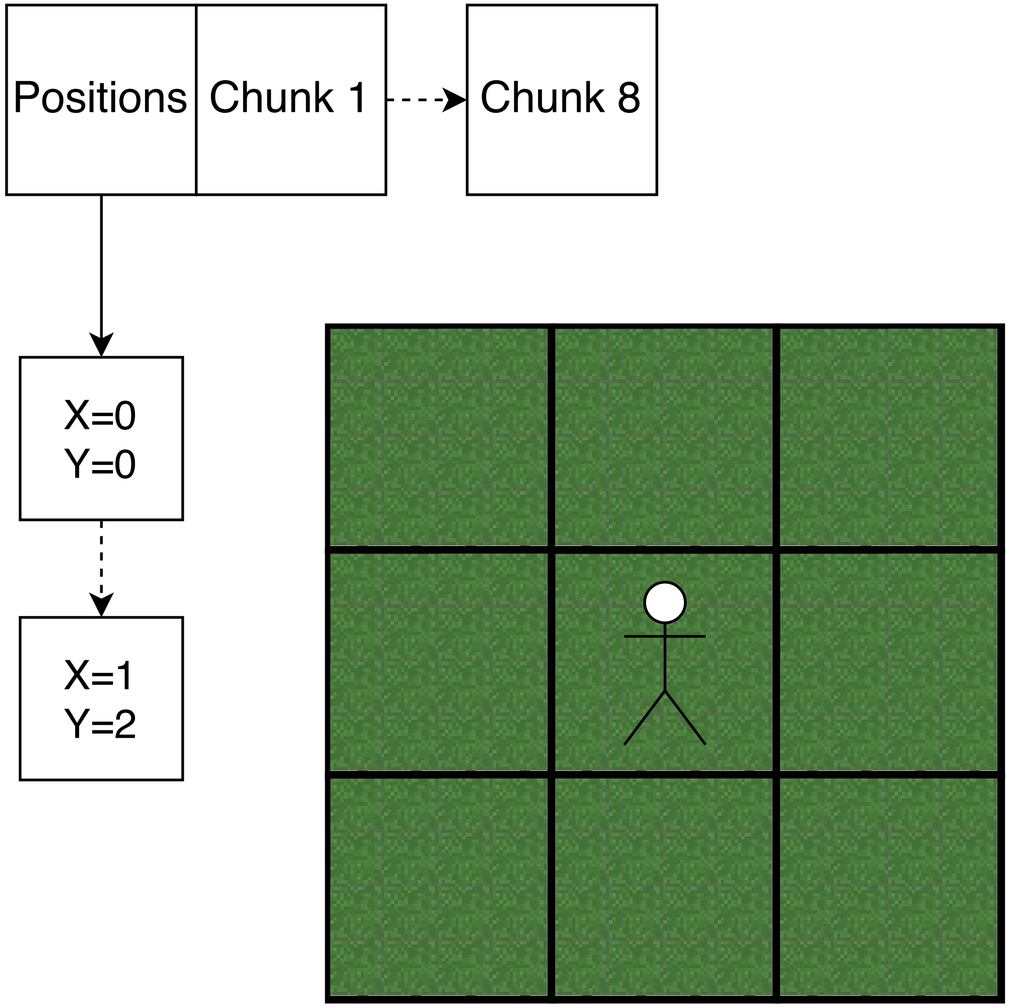
World Saving



Adding a new chunk

```
'packR'⎕CY'dfns'
pair←(2 2)
blocks← ⍴ (CHUNK DATA HERE)
⍴ blocks added to map
mapi,←(2*16)⊥pair
(packR blocks)⎕FAPPEND map
mapi ⎕FREPLACE(map ⋄ 1)
```

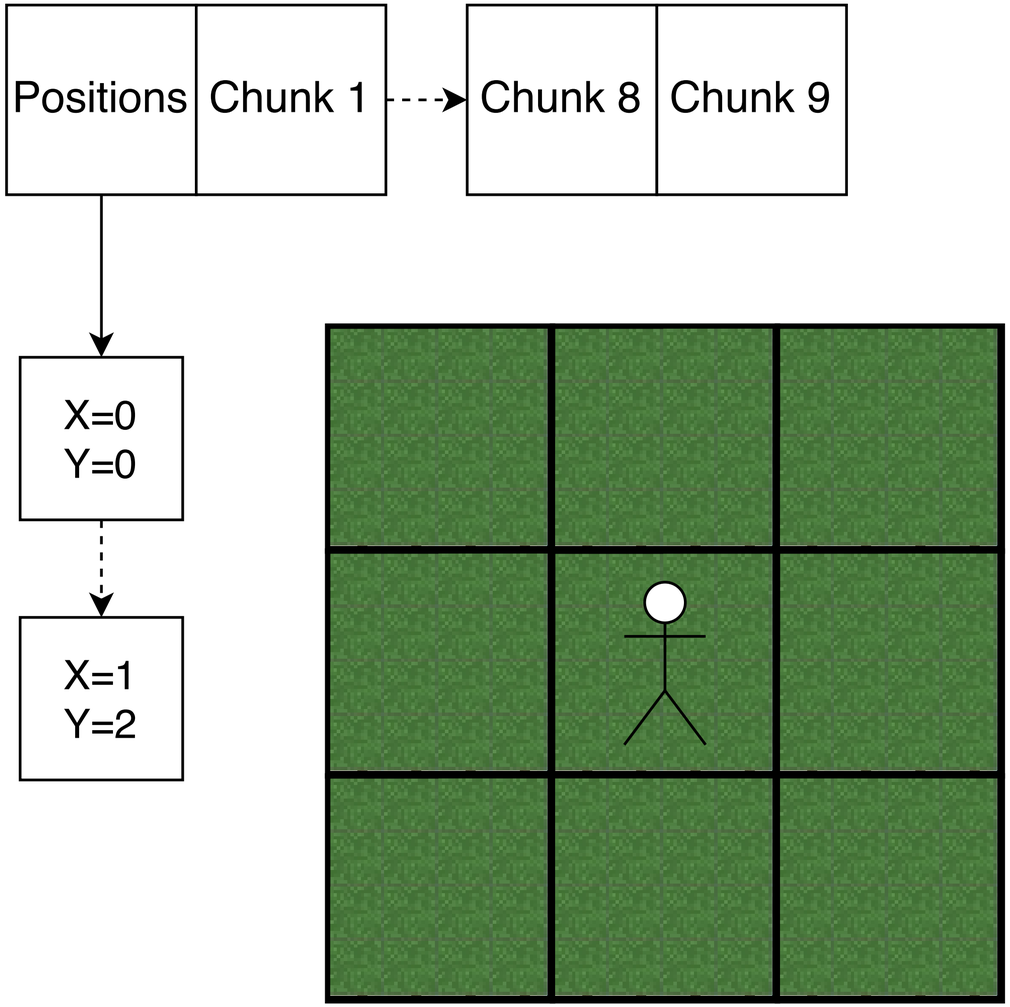
World Saving



Adding a new chunk

```
'packR'⎕CY'dfns'
pair←(2 2)
blocks← ⍬ (CHUNK DATA HERE)
⍬ blocks added to map
mapi,←(2*16)⊥pair
(packR blocks)⎕FAPPEND map
mapi ⎕FREPLACE(map ◇ 1)
```

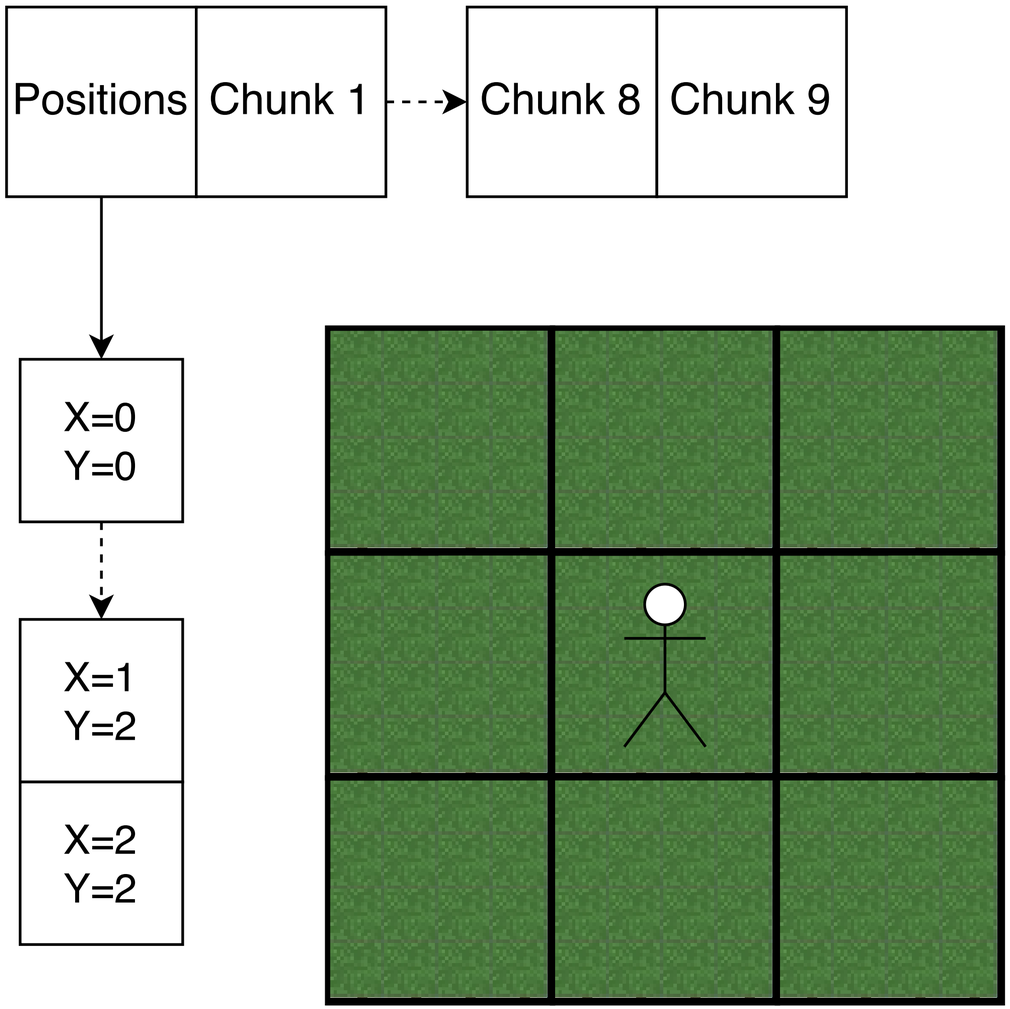
World Saving



Adding a new chunk

```
'packR'⎕CY'dfns'
pair←(2 2)
blocks← ⍬ (CHUNK DATA HERE)
⍬ blocks added to map
mapi,←(2*16)⊥pair
(packR blocks)⎕FAPPEND map
mapi ⎕FREPLACE(map ◇ 1)
```

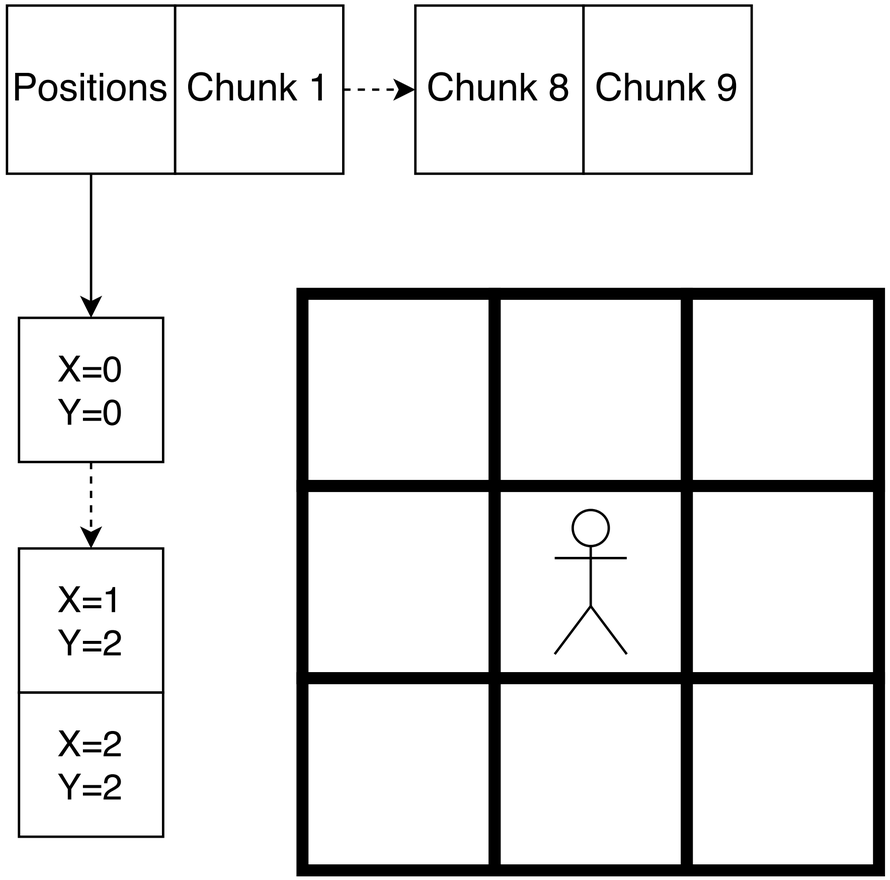
World Saving



Adding a new chunk

```
'packR'⎕CY'dfns'
pair←(2 2)
blocks← ⍬ (CHUNK DATA HERE)
⍬ blocks added to map
mapi,←(2*16)⊥pair
(packR blocks)⎕FAPPEND map
mapi ⎕FREPLACE(map ◇ 1)
```

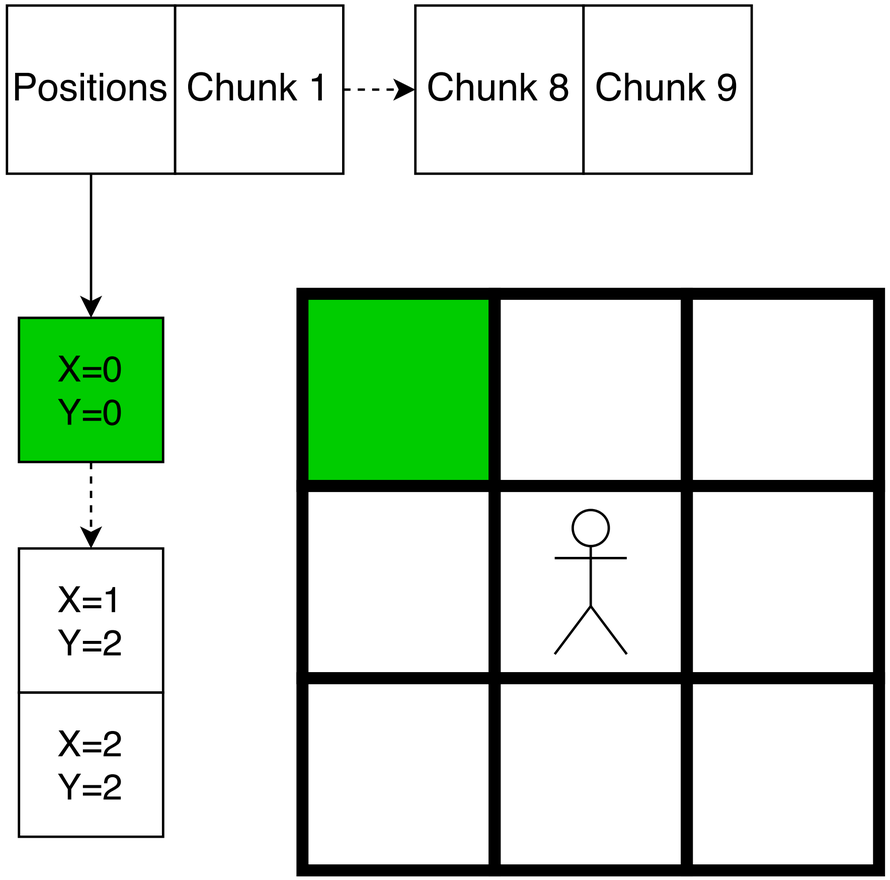
World Saving



Load chunks from component file

```
m←mapi⍷⍨(2*16)⊥⍤new
coords←⊃↓m⌿new
:For pair :In coords
    idx←(2*16)⊥pair
    blocks←0 packR ⎕FREAD(map ⋄ 2+mapi⍳idx)
    ⍝ Do stuff with blocks here
:EndFor
```

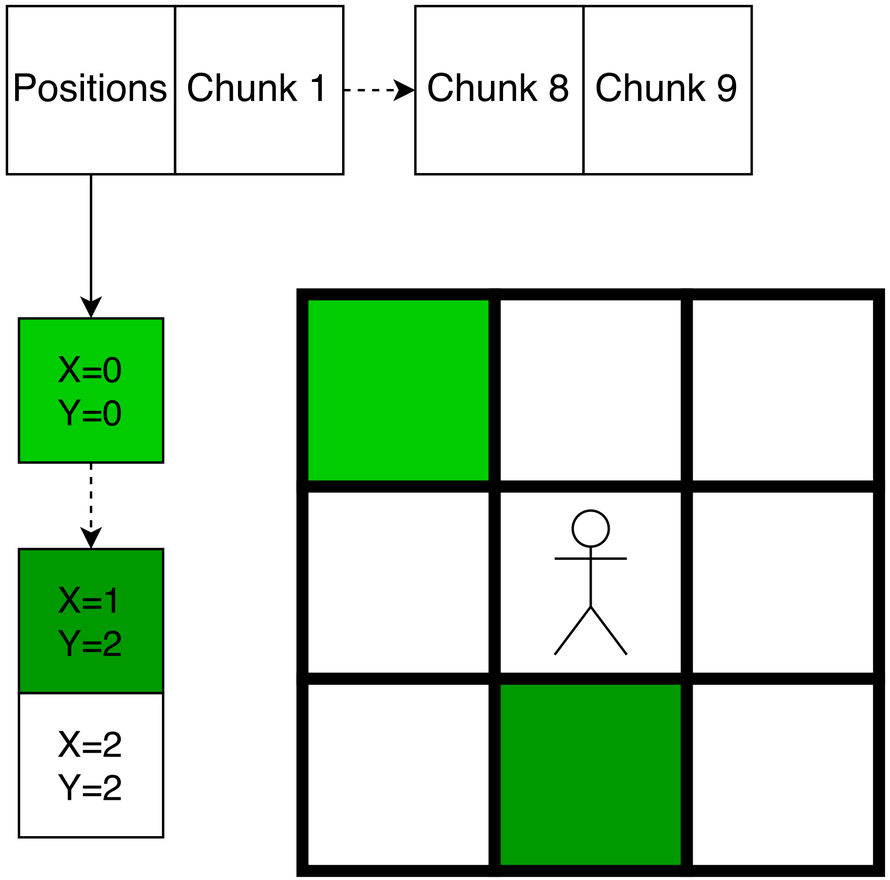
World Saving



Load chunks from component file

```
m←mapi∈⍨(2*16)⊥⍤¯1 new
coords←⊃↓m⌿new
:For pair :In coords
    idx←(2*16)⊥pair
    blocks←0 packR ⎕FREAD(map ⋄ 2+mapi⍳idx)
    ⍝ Do stuff with blocks here
:EndFor
```

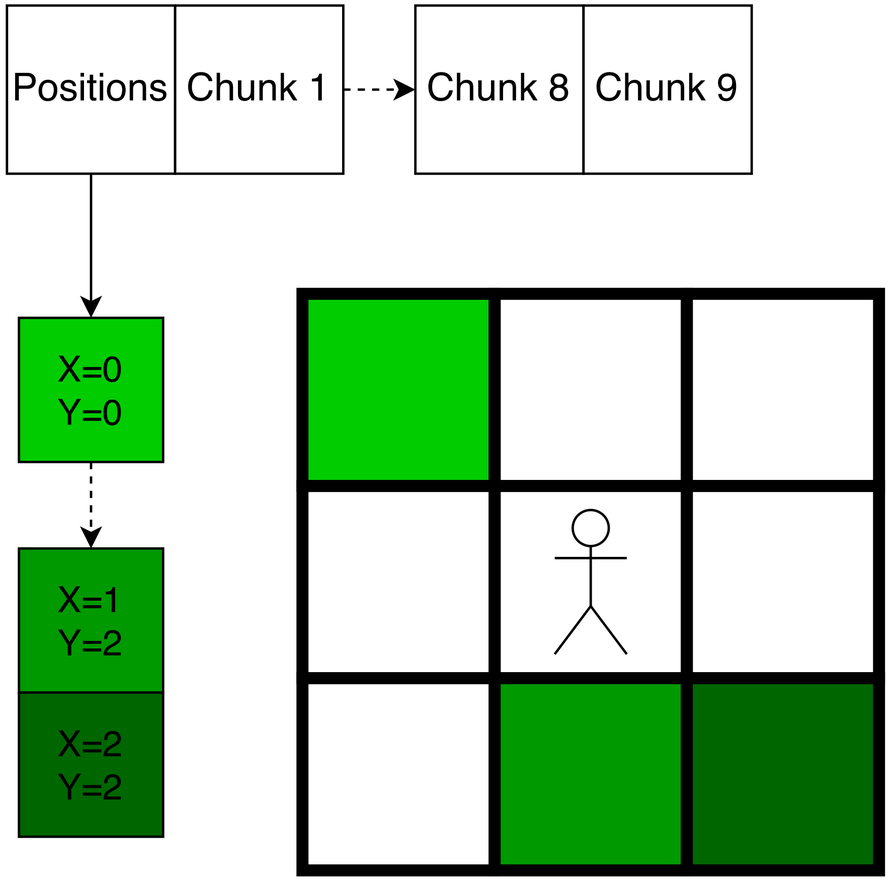
World Saving



Load chunks from component file

```
m←mapi∈⍨(2*16)⊥⍤¯1⍤new
coords←⊃↓m⌿new
:For pair :In coords
    idx←(2*16)⊥pair
    blocks←0 packR ⎕FREAD(map ⋄ 2+mapi⍳idx)
    ⍝ Do stuff with blocks here
:EndFor
```

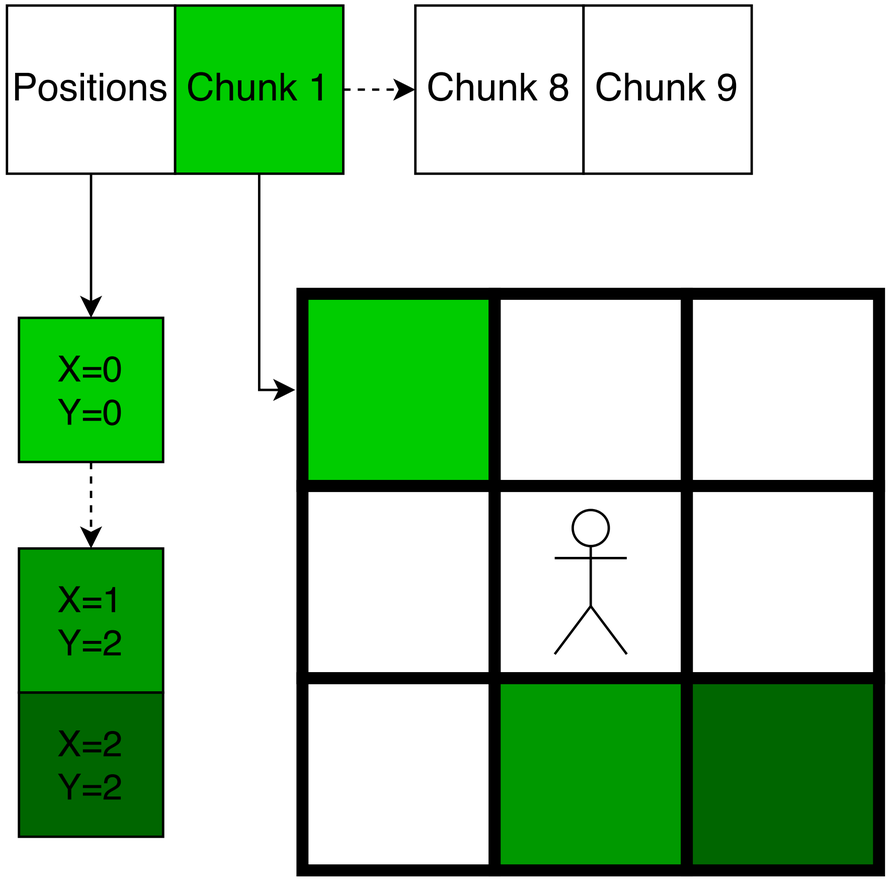
World Saving



Load chunks from component file

```
m←mapi⍷⍨(2*16)⊥⍤new
coords←⊃↓m⌿new
:For pair :In coords
    idx←(2*16)⊥pair
    blocks←0 packR ⎕FREAD(map ⋄ 2+mapi⍳idx)
    ⍝ Do stuff with blocks here
:EndFor
```

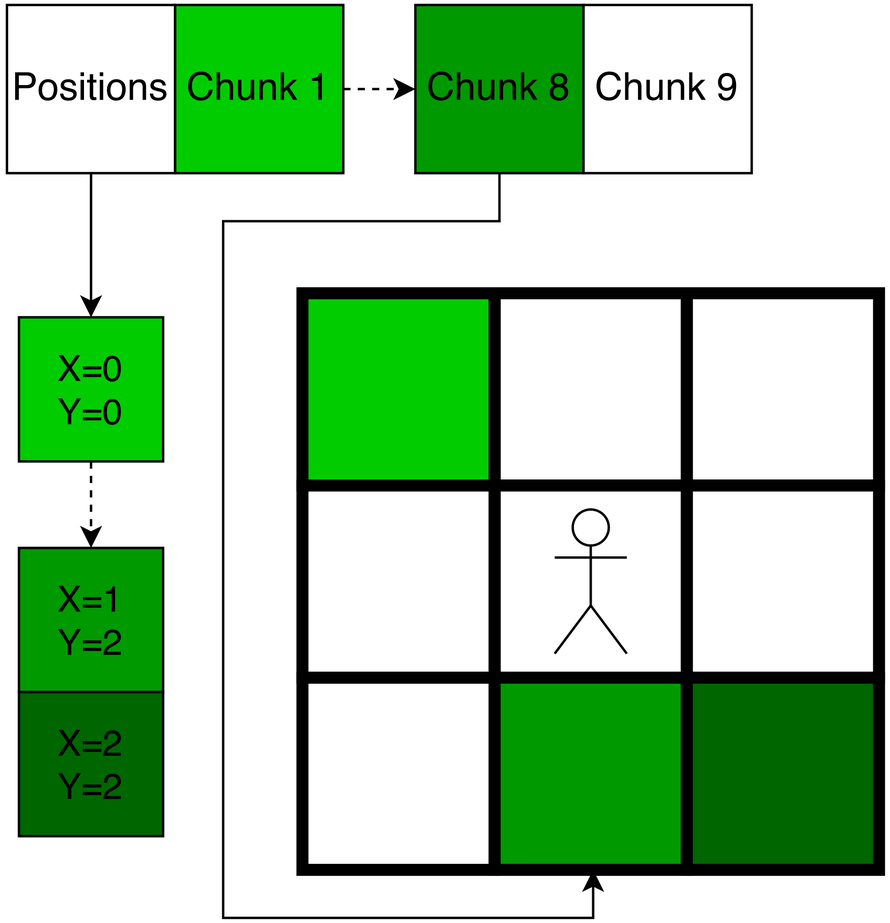
World Saving



Load chunks from component file

```
m←mapi∈⍨⍤(2*16)⊥⍤¨new
coords←⊃↓m⌿new
:For pair :In coords
    idx←(2*16)⊥pair
    blocks←0 packR ⎕FREAD(map ⋄ 2+mapi⍳idx)
    ⍝ Do stuff with blocks here
:EndFor
```

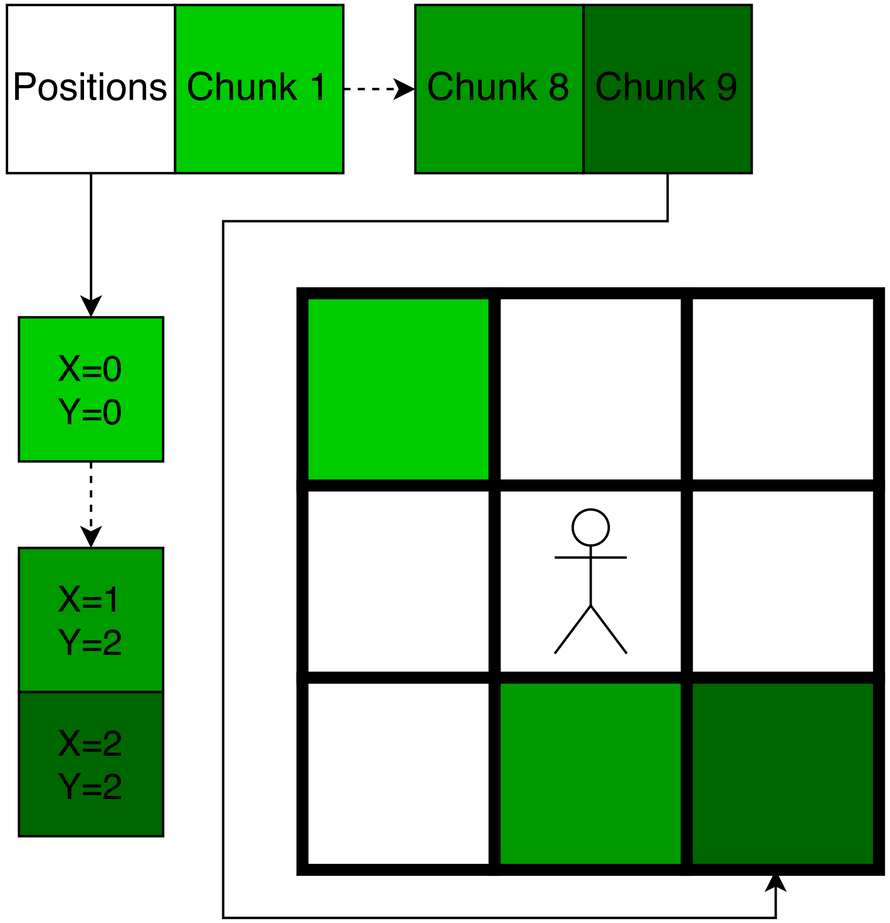
World Saving



Load chunks from component file

```
m←mapi∈⍨(2*16)⊥⍤new
coords←⊃↓m⌿new
:For pair :In coords
    idx←(2*16)⊥pair
    blocks←0 packR ⎕FREAD(map ⋄ 2+mapi⍳idx)
    ⍝ Do stuff with blocks here
:EndFor
```

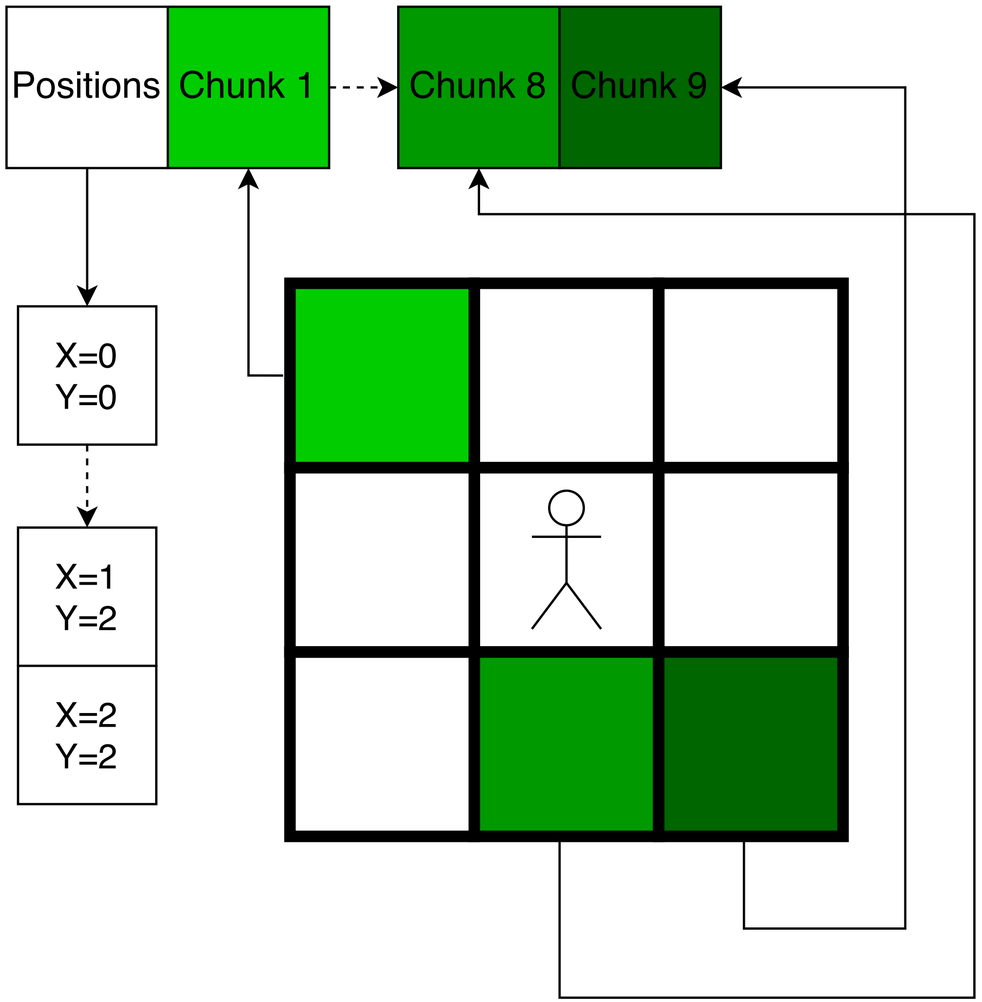
World Saving



Load chunks from component file

```
m←mapi⍷⍨(2*16)⊥⍤new
coords←⊃↓m⌿new
:For pair :In coords
    idx←(2*16)⊥pair
    blocks←0 packR ⎕FREAD(map ⋄ 2+mapi⍳idx)
    ⍝ Do stuff with blocks here
:EndFor
```

World Saving



Saving all chunks on exit

```
(⍳≠chunks){
    (packR α⌷chunks)⌻FREPLACE(map ⋄ 2+⍵)
}¨mapi⍳Cg'enc_xz'
```

Why should I care?

Why should I care?

Evidently…

Rank	Title	Sales (millions)	Series	Platform(s)	Release year	Developer(s)	Publisher(s)	Ref.
1[a]	*Tetris*	520	None	Multi-platform	1988[c]	Various	Various	[17]
	Minecraft	400	*Minecraft*	Multi-platform	2011[d]	Mojang Studios	Mojang Studios	[20]
2	*Grand Theft Auto V*	230	*Grand Theft Auto*	Multi-platform	2013	Rockstar North	Rockstar Games	[21]
3	*Red Dead Redemption 2*	87	*Red Dead*	Multi-platform	2018	Rockstar Games	Rockstar Games	[21]
4	*Wii Sports*[b]	82.9	*Wii*	Wii	2006	Nintendo EAD	Nintendo	[24]
5	*Mario Kart 8* / *Deluxe*	79.99	*Mario Kart*	Wii U / Switch	2014	Nintendo EAD / Nintendo EPD (*Deluxe*)	Nintendo	[e]

Why should I care?

- Games are a good benchmark
  - Good performance is required
  - Need to be able to iterate
  - Games are a giant ball of mutable state
- APL shows promise at being a “tool of thought” in this area!
  - Nearly all primitives used
  - ~1,700 lines of APL

Questions?