aws FOR GAMES

How to generate Pixel Perfect 2D levels in Godot 4

with the help of Generative AI

Mat Rowlands

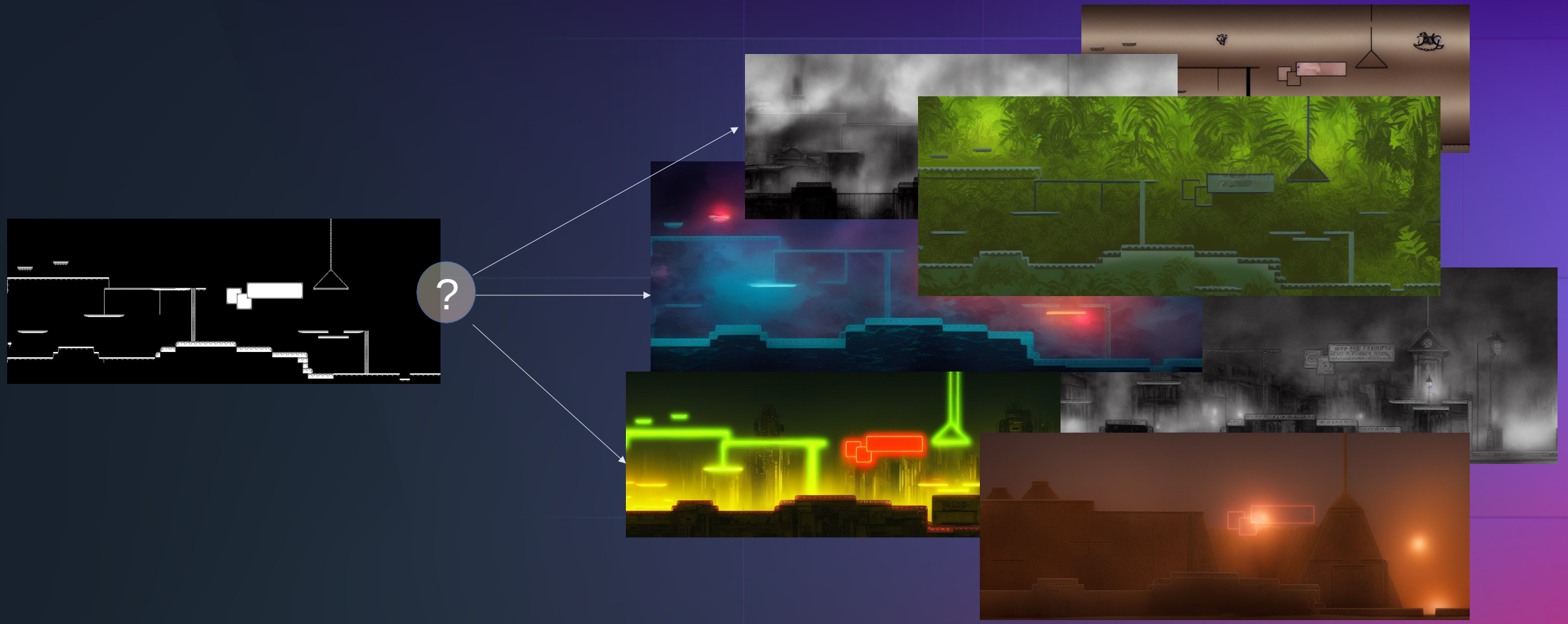
Senior Game Tech Solutions Architect

Who am I?

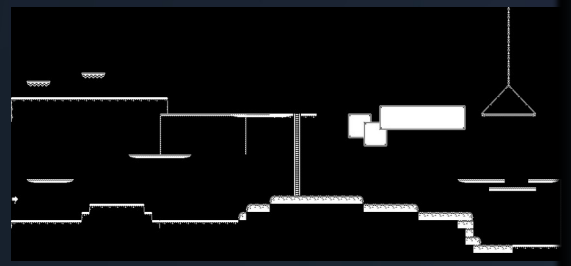
- AWS Game Tech Solution Architect
- 3D worlds and Machine Learning
- Hardware tinkerer
- Esoteric retro game launcher
- Lots of records
- … Lots more tours

And a truly terrible artist...

Going from icky to ahhhh

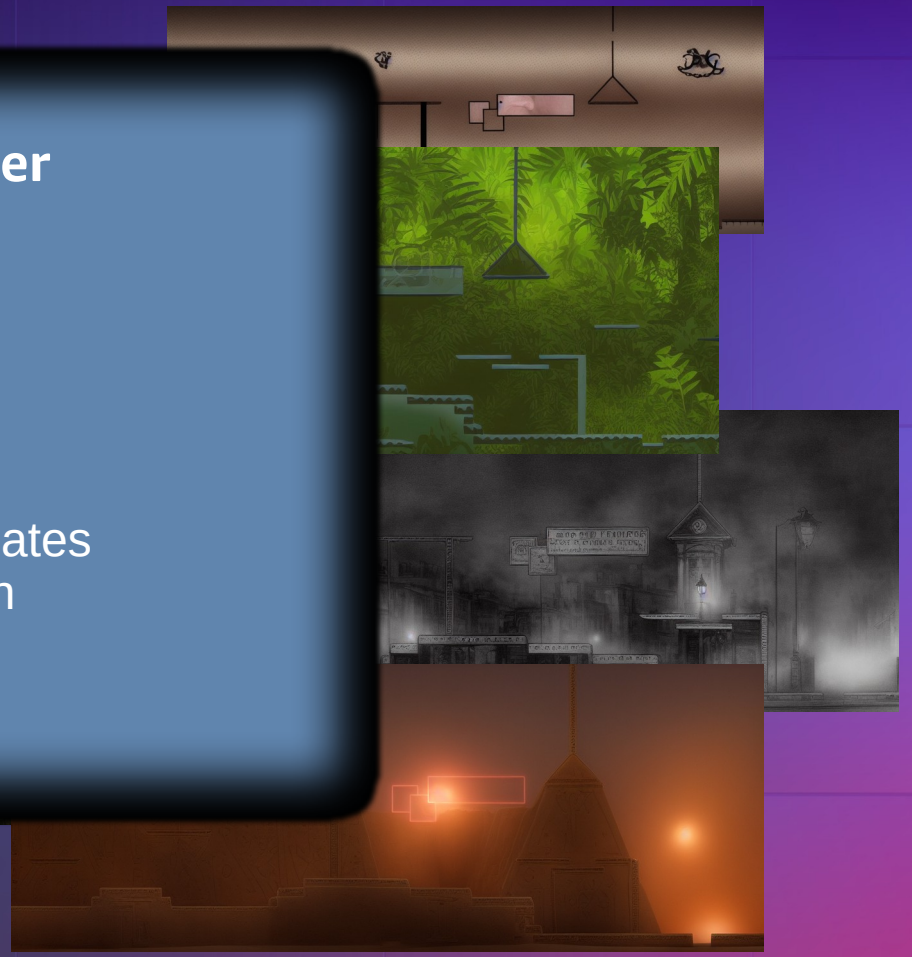


Going from icky to ahhhh

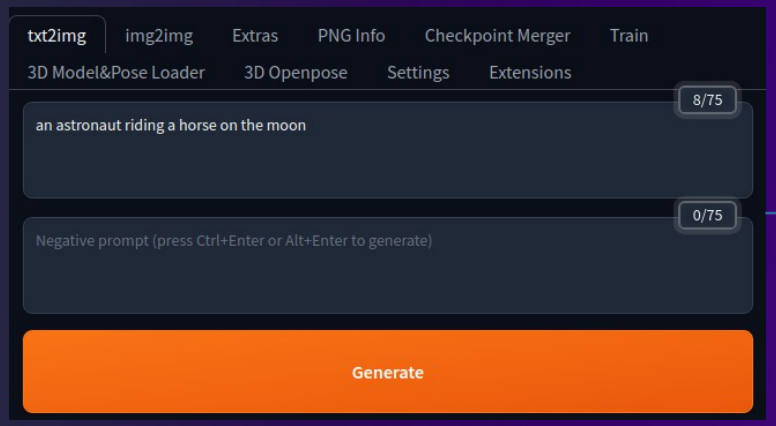


What we're going to cover

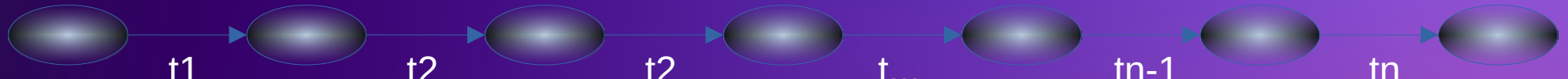
- What changed in tech
  - Stable Diffusion
  - ControlNet
  - Automatic111 Interface
- How to set up Godot 4
- How to create Level Templates
- How to set up the toolchain
- Demo
- Tips



Diffusion Models



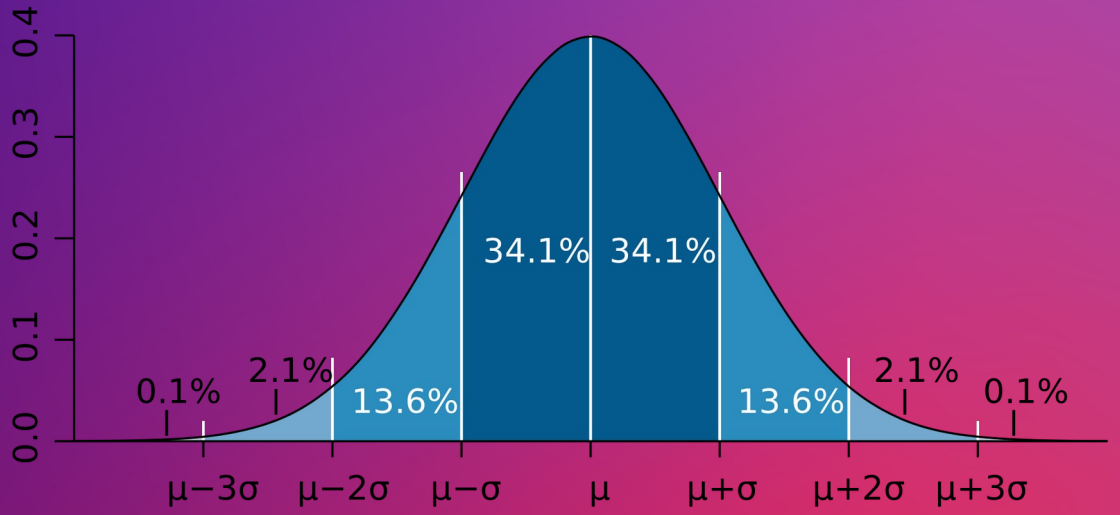
Diffusion Process - Training



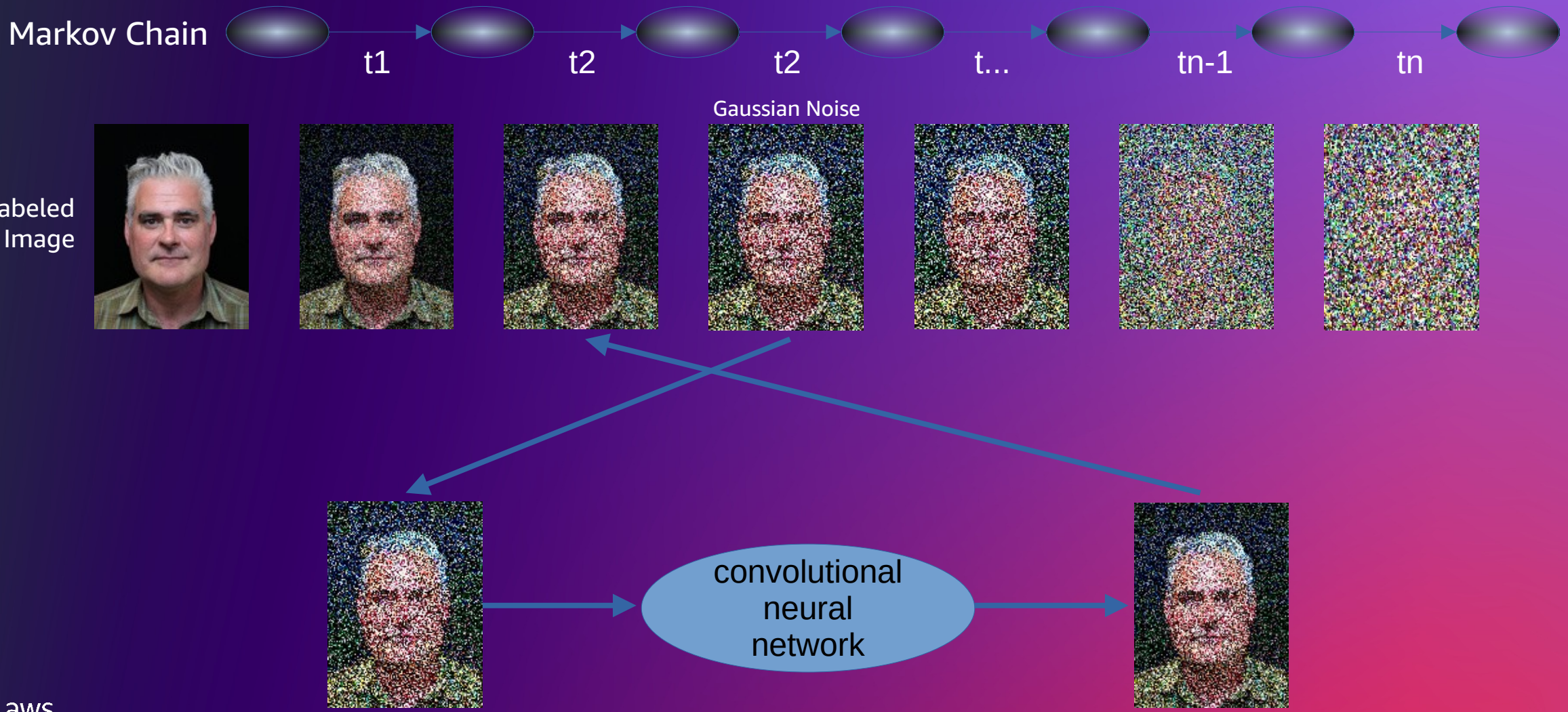
Gaussian Noise

Labeled Image

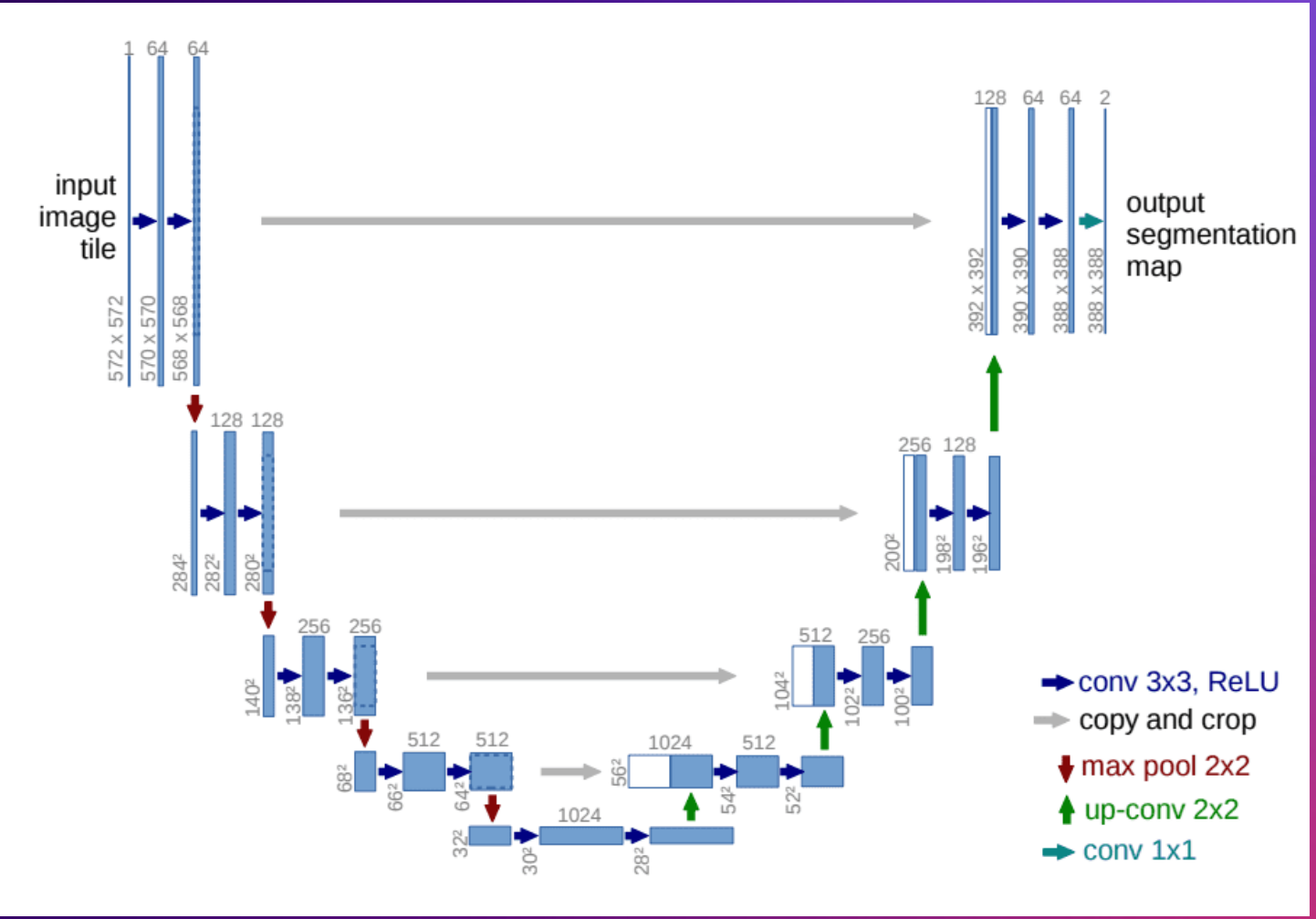
A Markov Chain is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event - *Wikipedia*



Diffusion Process - Training

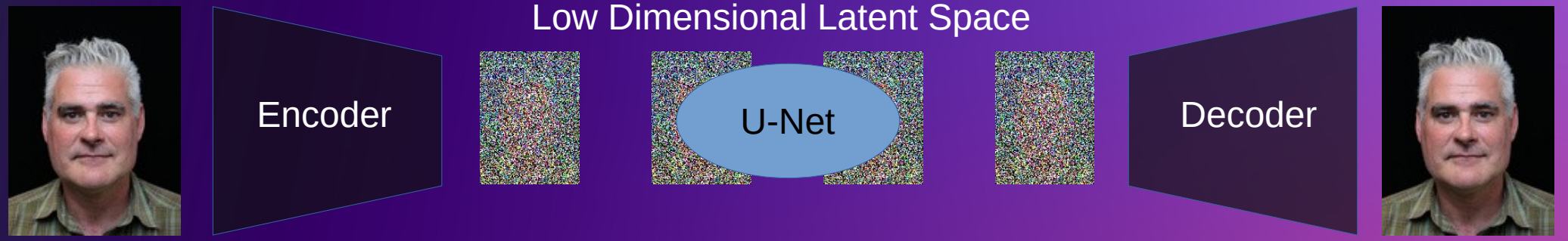


Diffusion Process - U-Net

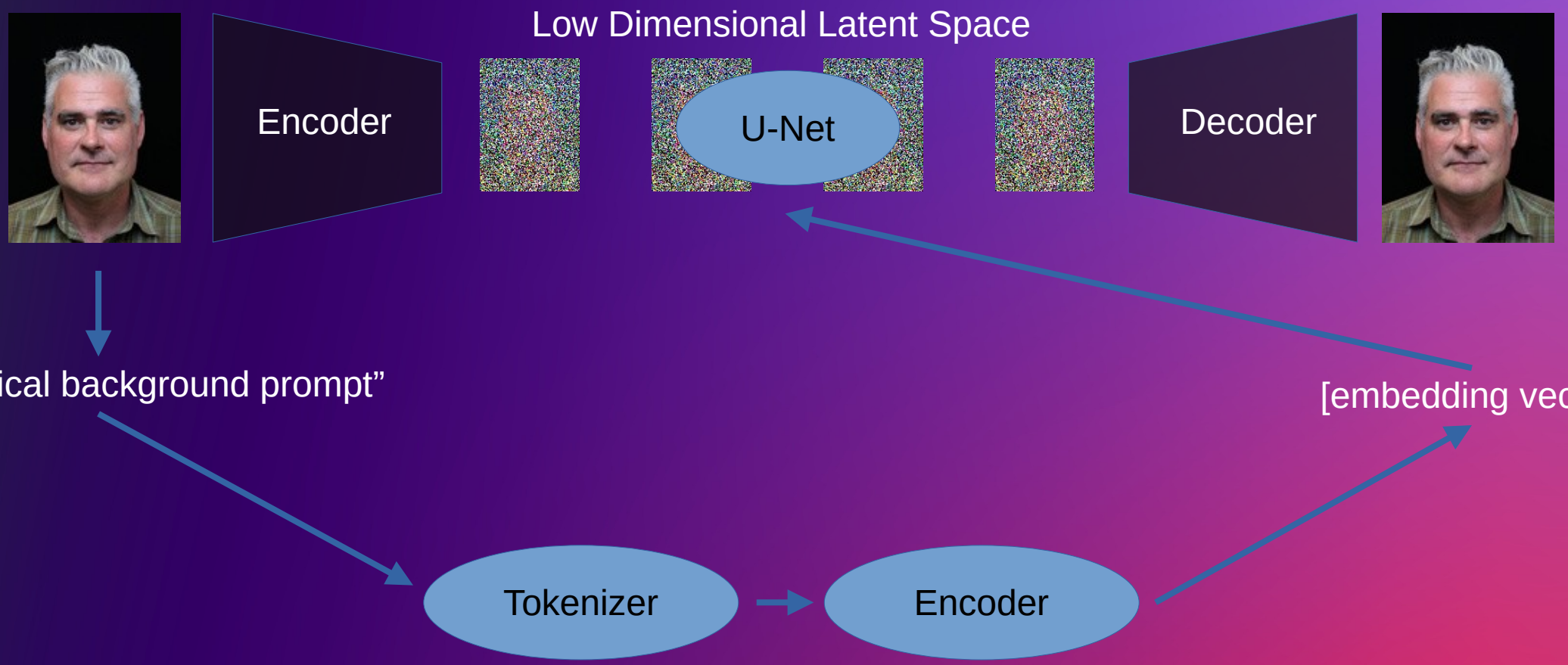


https://arxiv.org/abs/1505.04597

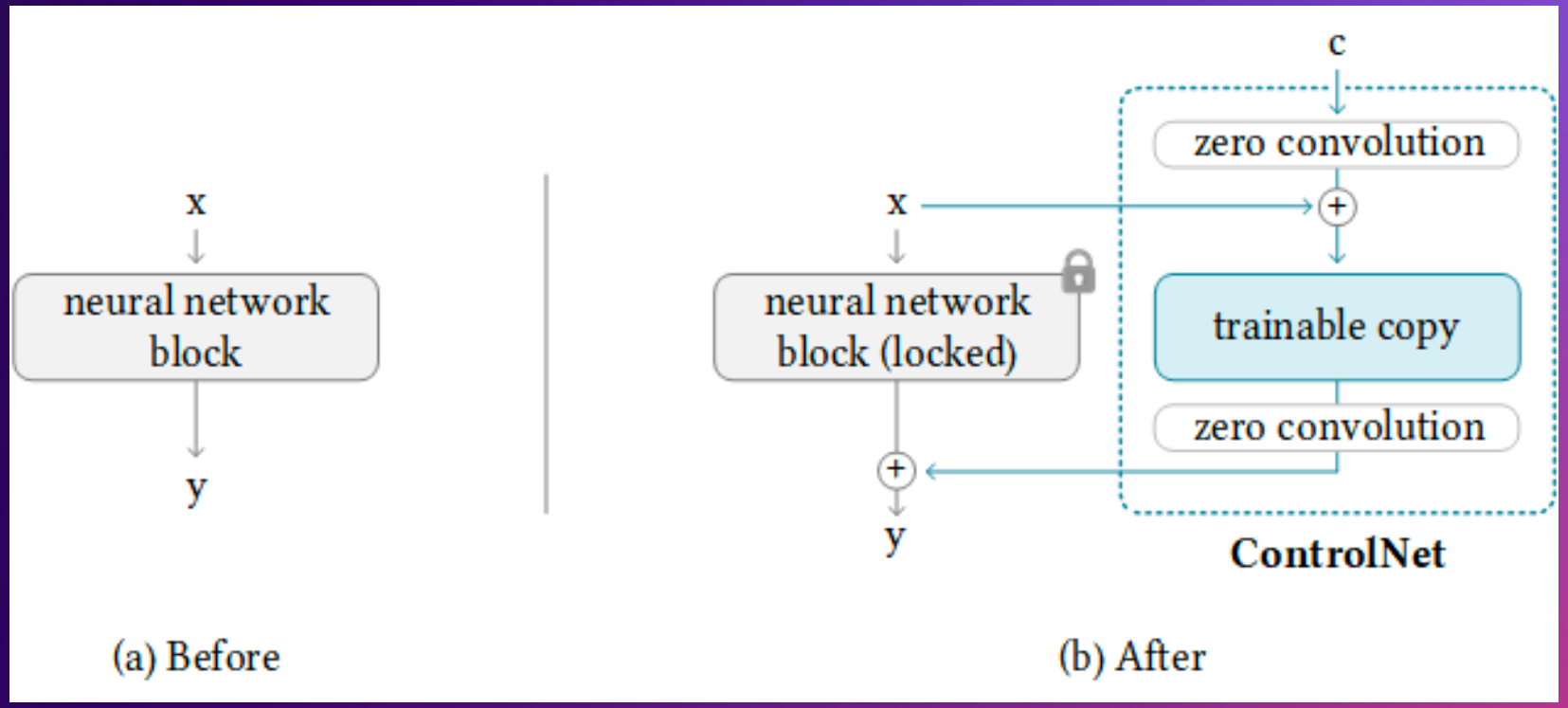
Diffusion Process - Pipeline



Diffusion Process - Pipeline



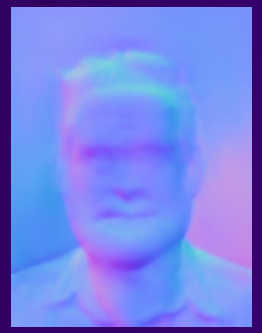
ControlNet



https://github.com/lllyasviel/ControlNet

https://arxiv.org/abs/2302.05543

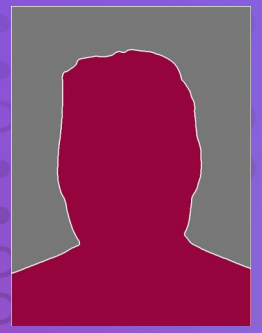
Pre-processing for ControlNet



Normal

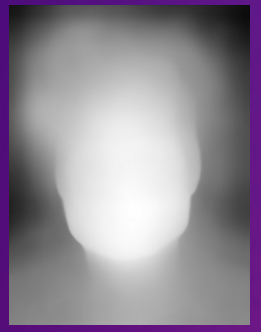
Line Art

Original



Segmentation

Canny



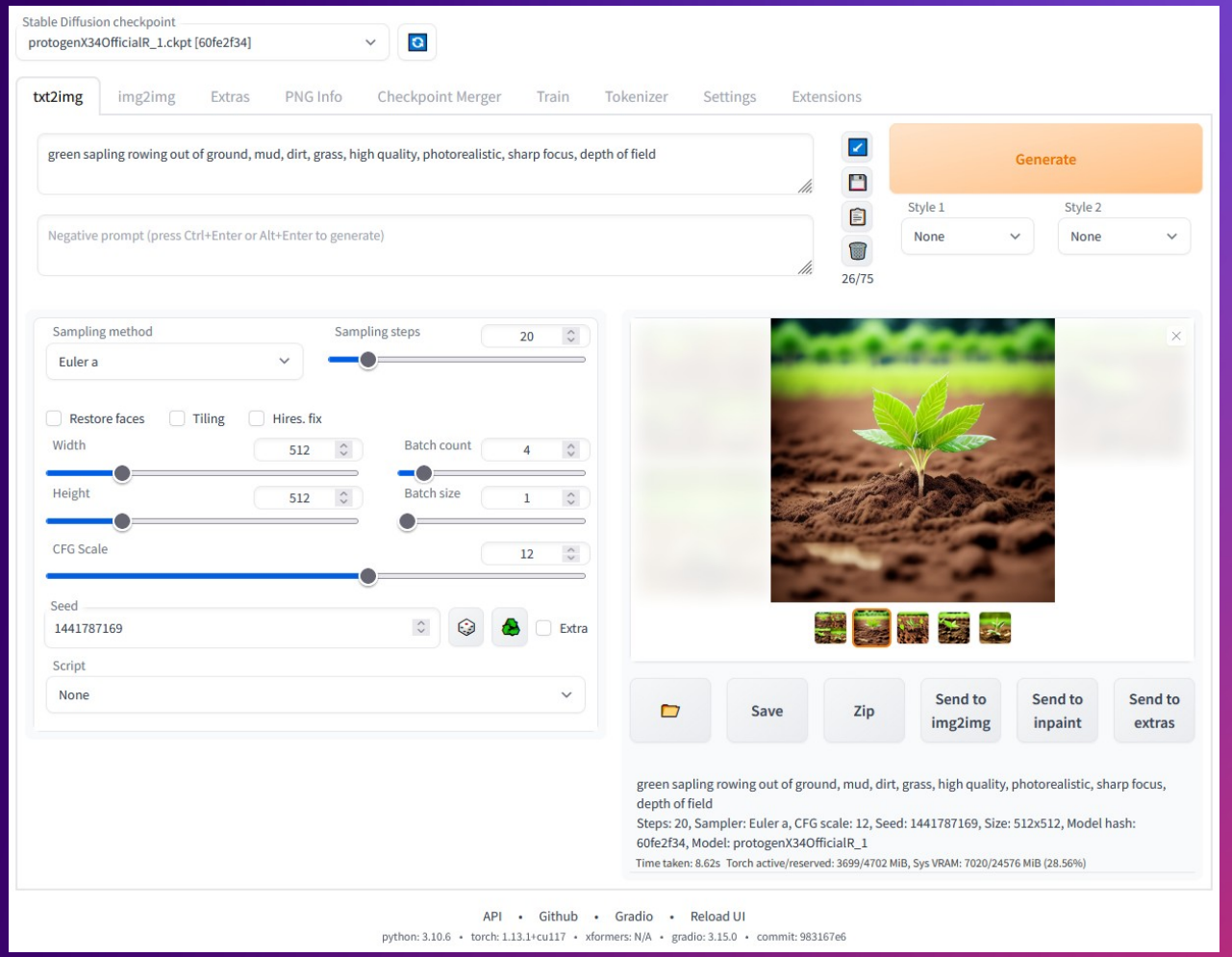
Depth

Soft Edge

Scribble

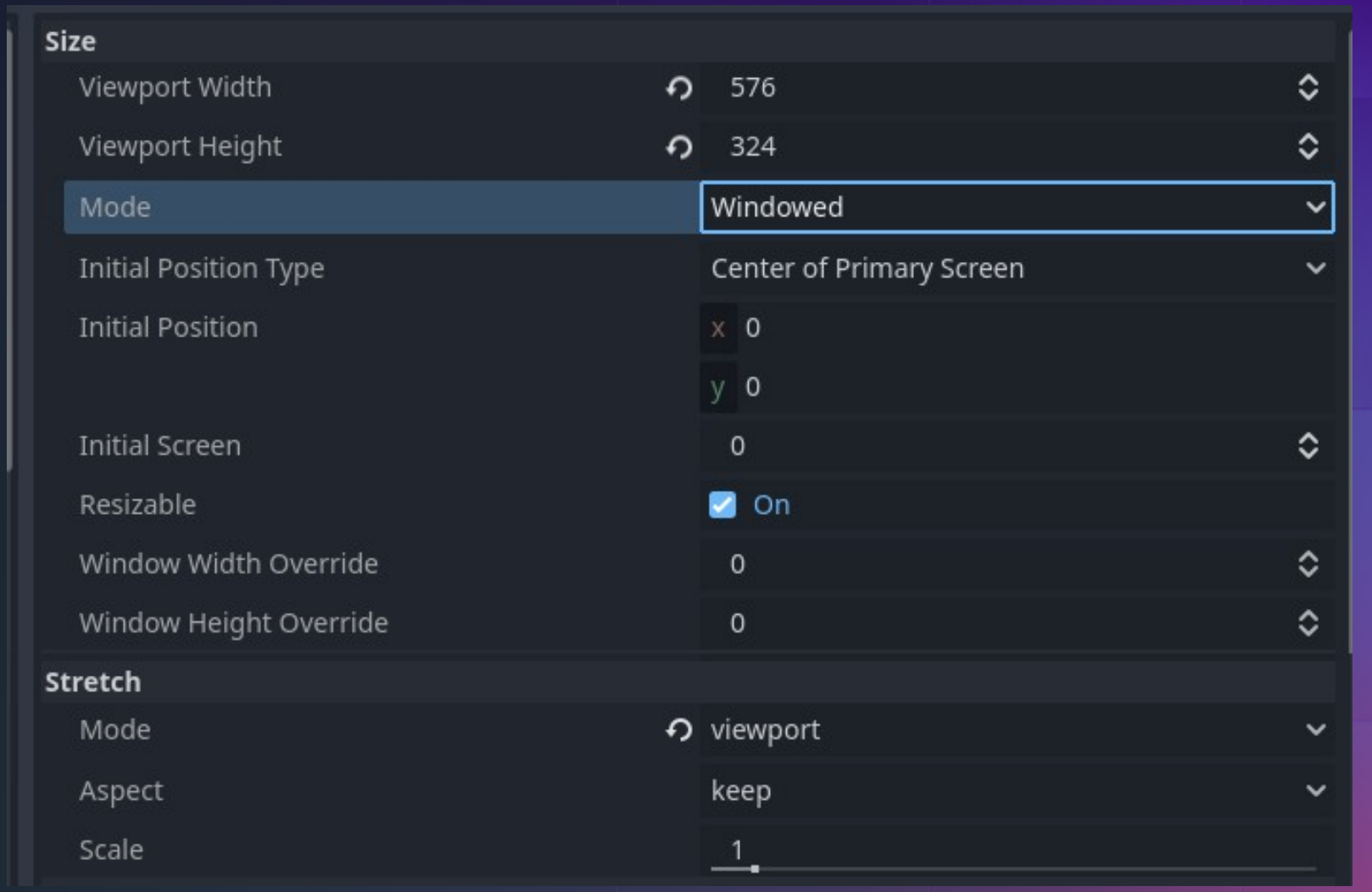
ControlNet + Stable Diffusion

Automatic111 for the People

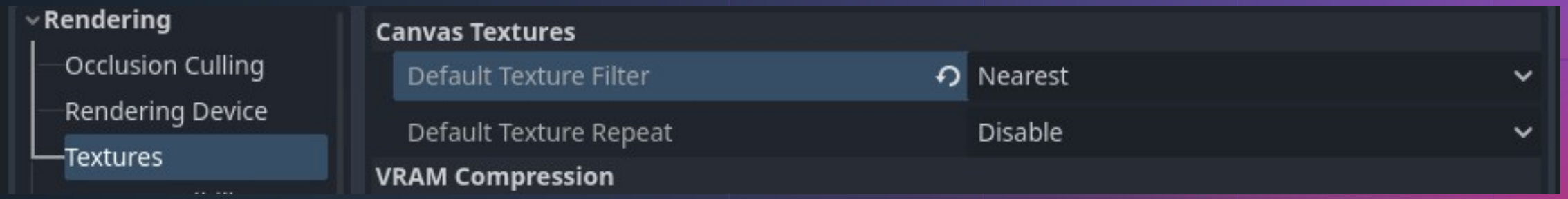


https://github.com/AUTOMATIC1111/stable-diffusion-webui

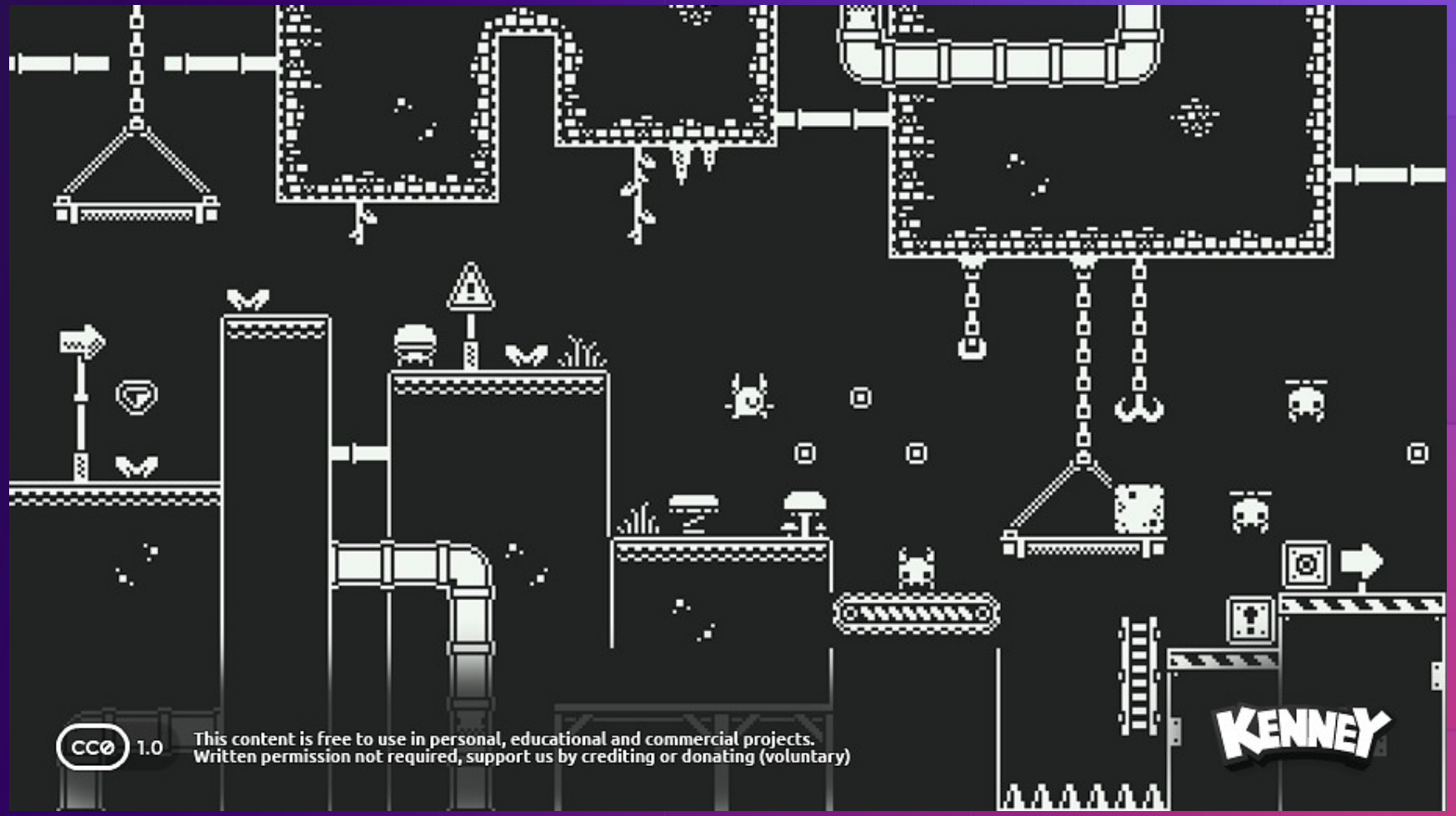
Setting up Godot 4 – Display Server



Setting up Godot 4 – Rendering & Snap

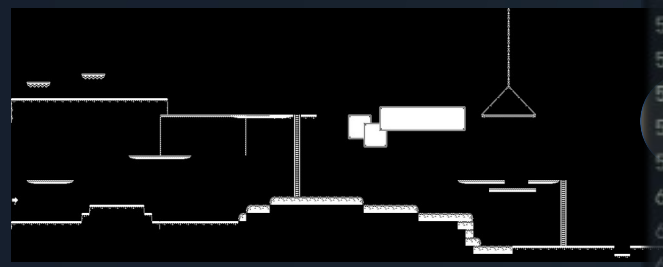


Level Design



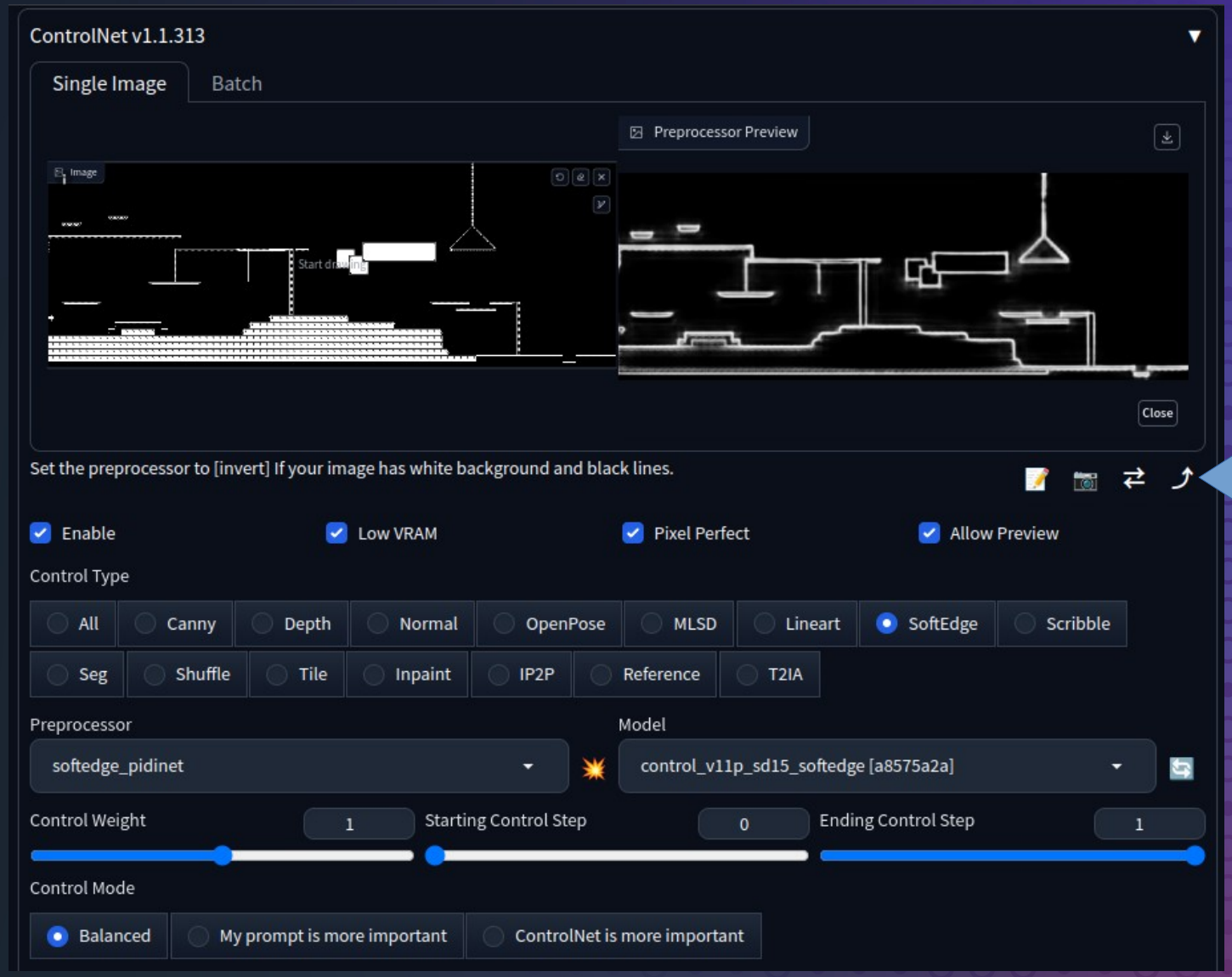
https://kenney.nl/assets/1-bit-platformer-pack

Level Design to Image Template



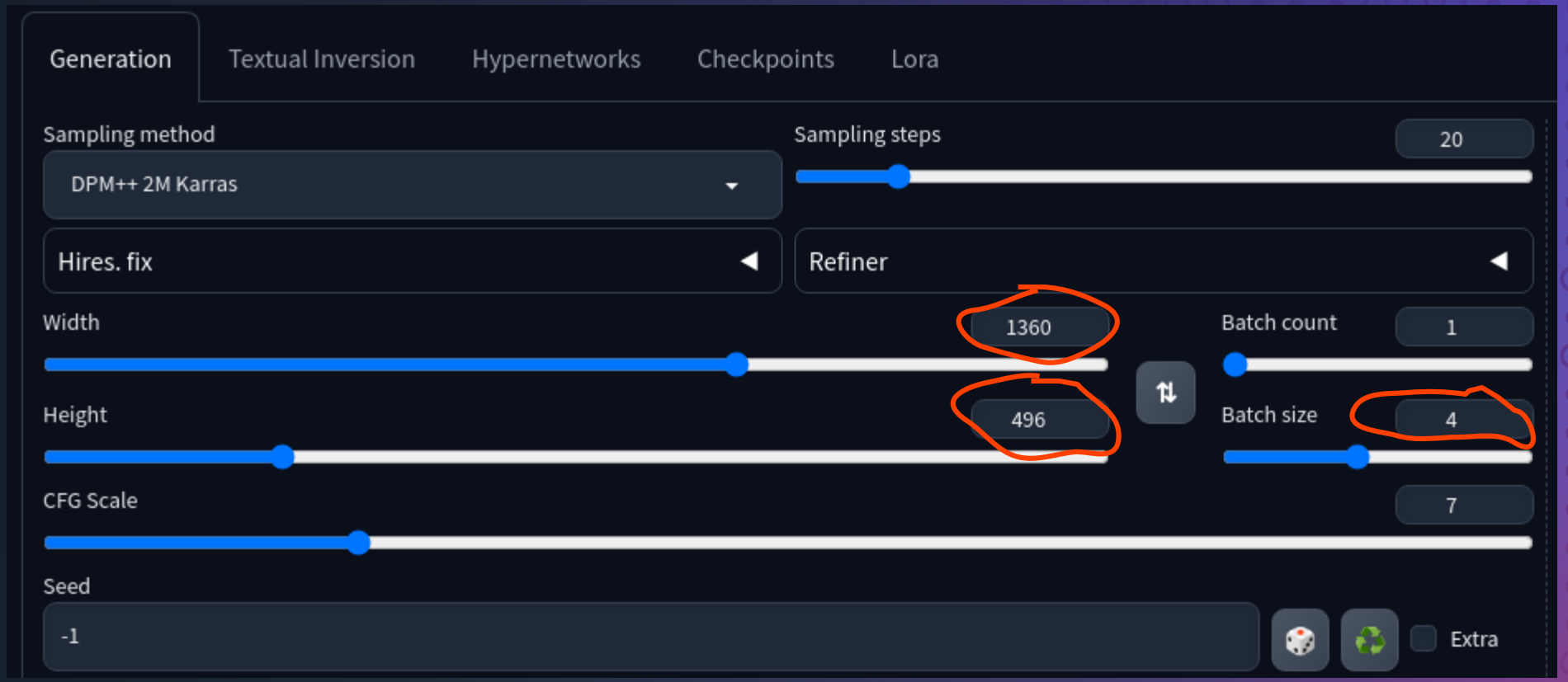
```gdscript
var sourceRect = get_viewport().get_visible_rect()

var moreY = true
var yStep = int(vp_max_y  / 5)
var xStep = int(vp_max_x / 5)

while moreY:
    map.position.y = current_y
    current_x = 0
    #print("map y = ", map.position.y)
    while current_x < max_x:
        map.position.x = -current_x
        #print("map x = ", map.position.x)
        map.can_process()
        await RenderingServer.frame_post_draw
        await get_tree().create_timer(0.05).timeout
        var chunk = get_viewport().get_texture().get_image()
        var destPos = Vector2i(current_x, abs(current_y))
        img.blit_rect(chunk, sourceRect, destPos)
        current_x = current_x + xStep

    current_y = current_y - yStep
    if current_y < -max_y - yStep:
        moreY = false

var template = "template-" +  parent.name + ".png"
var dir = DirAccess.open(".")
if dir:
    if not dir.dir_exists("templates"):
        dir.make_dir("templates")

img.save_png("./templates/" + template)
```

Importing our Template



Copy template dimensions

Generate your level backgrounds



Demo

Tips for generating level art

- Use chained ControlNets
  - Settings / ControlNet / Multi ControlNet: Max models amount
  - Go from widest to narrowist. eg Scribble → SoftEdge
    - Set the Control Steps to move between them
      - Ending Control Step to 0.5 on #1
      - Starting Control Step to 0.5 on #2
- Use Semantic Segmentation to get to a block pallette, then Canny
- Use a refiner to add style, Switch at step 0.8 or higher
- The same seed with the same prompt will give the same result
  - The seed can be found in the log for a generated image
  - Use this to iterate on a style

Tips for godot templates

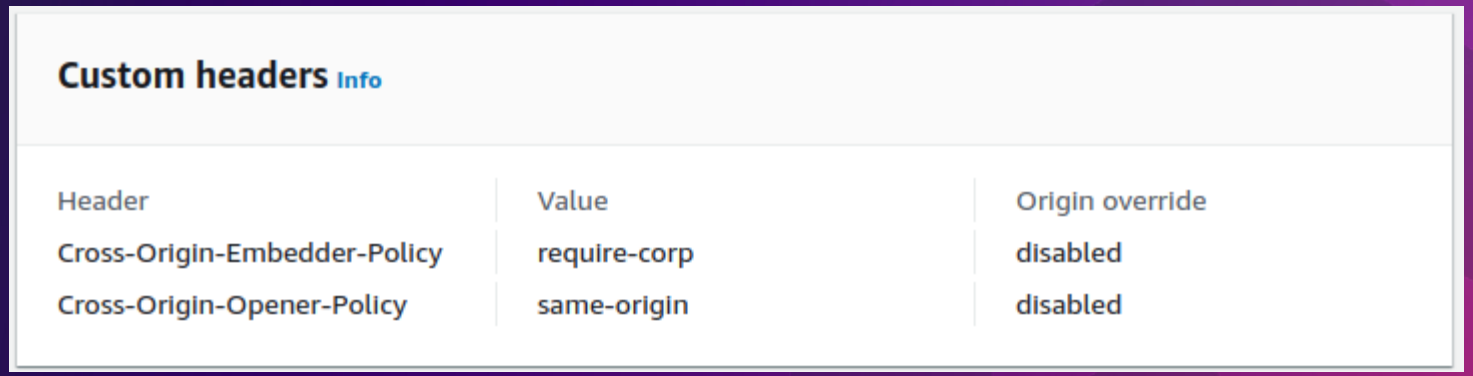
- Use TileMap layers, only generate a template with the static segments
- Put a hard black ColorRect behind the Map for maximum contrast
- Modulate the moving tiles layer on top based on inverse dominant color

```
var rnd : int = randi_range(0, len($TileMap.get_used_cells(0)) – 1)
var cell : Vector2i = $TileMap.get_used_cells(0)[rnd]
var pix_x : int = (cell_size.x * cell.x) + (cell_size.x / 2)
var pix_y : int = (cell_size.y * cell.y) + (cell_size.y / 8)
$TileMapLevel1.modulate(image.get_pixel(pix_x, pix_y).inverted())
```

- Use pixel grid snapping
- Viewport stretch mode
- Experiment with block filling enclosed areas
- Use simple tilesets

Tips for AWS

- Use a g5.2xlarge for Automatic111, shut it down when not needed
- Host your game with no infrastructure
  - Set up Export to web as a target, and export to a folder
  - Set up an S3 Bucket as a static website
  - Run : aws s3 sync ./export/folder s3://your-bucket –acl public-read
  - Create a Cloudfront CDN distribution in front of the S3 bucket
  - Set these custom headers for the Response Header Behaviour



aws FOR GAMES

Thank you!

Mat Rowlands

matrow@amazon.de

Demo : https://gic.matrow.sa.aws.dev

Repo : https://www.github.com/podulator/gic-2023