NARVAL

GAME ENGINE

Desenvolvimento da game engine 2D Narval utilizando OpenGL e Java

Igor Batista Fernandes
Orientadora Prof.ª Dr.ª Maria Adriana Vidigal de Lima

Game Engine

Game engine é um sistema composto de outros sub sistemas cujo trabalho em coesão provêm as funcionalidades necessárias para o desenvolvimento de de um jogo.

Seus principais sub sistemas operam e implementam as seguintes funcionalidades:

- Renderização
- Colisões e Física dos objetos
- Aúdio
- IA

Game Loop

Game Loop

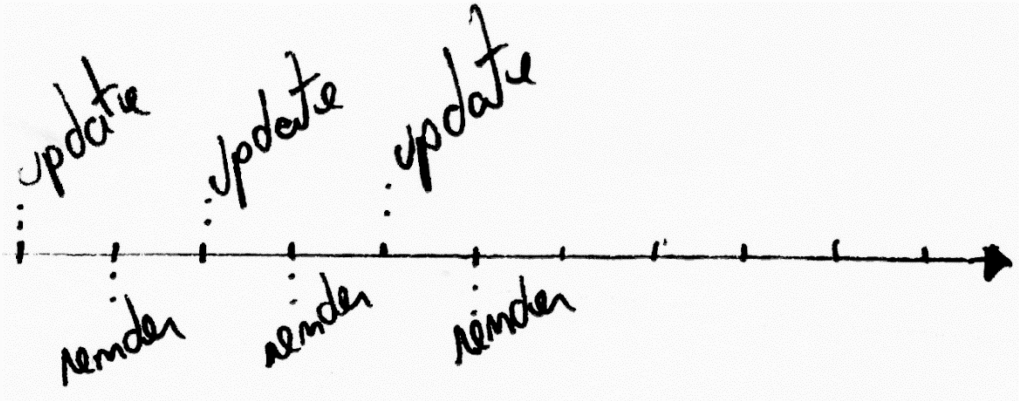
É o núcleo de toda game engine. Nele está o *while(running)* que invoca os métodos *update()* e *render()*.

Para o método *update()* existem três principais formas:

- Timestep fixo
- Timestep variável
- Timestep semifixo

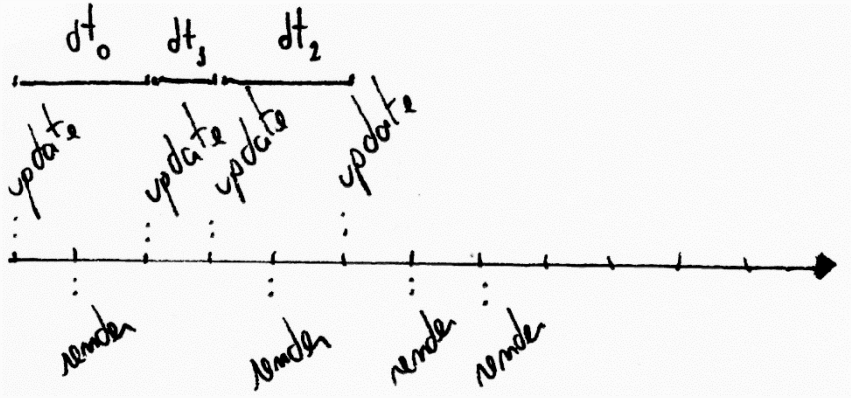
Game Loop de **Timestep fixo**

- Cada *tick* (Δt) no relógio que representa o tempo do mundo no jogo é previamente fixado.
- Valores comuns são de 30 ou 60 chamadas por segundo (33ms e 16ms, respectivamente). Entretanto, é uma decisão específica de cada projeto.



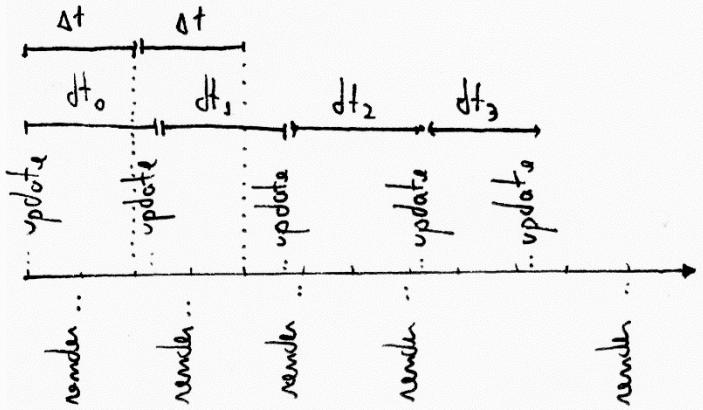
Game Loop de **Timestep variável**

- Cada *tick* (Δt) no relógio que representa o tempo do mundo no jogo é calculado em tempo real. Isso desacopla o método *udpate*, permitindo ser executado quantas vezes possível.
- O valor Δt é obtido calculando-se quanto tempo se passou entre a chamada anterior e a atual, repassando essa diferença aos sub sistemas.



Game Loop de **Timestep semifixo**

- De maneira análoga ao timestep fixo, define-se um valor estático para Δt . Entretanto, se por ventura o sistema levar mais que Δt entre um *update* e outro ele realiza um *catch-up,* chamando *update* até que o sistema esteja novamente sem atrasos. Isso é realizado através de uma variável acumuladora.

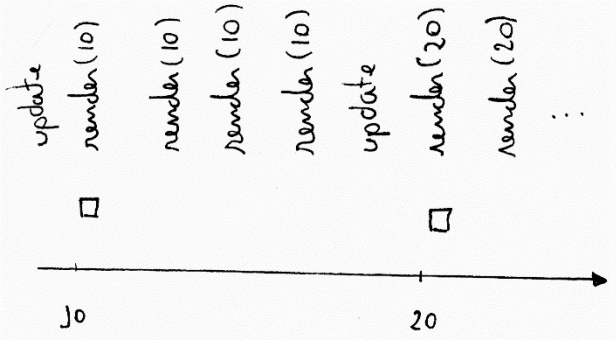


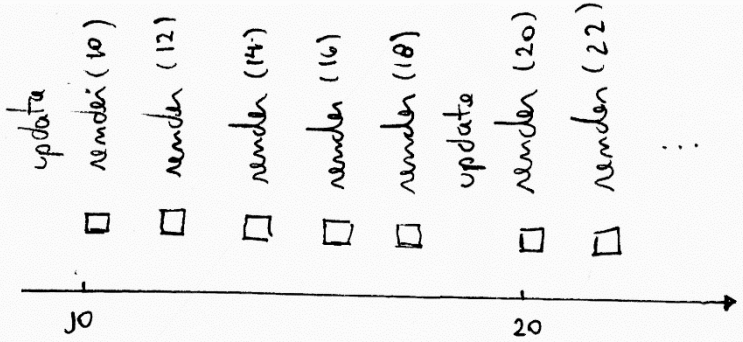
Stuttering (Engasgamento)

- O efeito de *stuttering* é causado em duas situações:
  - 1. Quando o sistema renderiza as posições dos objetos usando diretamente os valores retornados pelo *update().*
  - 2. Quando o sistema precisa realizar *catch-ups* frequentemente, ou seja, a máquina não possui poder de processamento suficiente para executar o jogo.

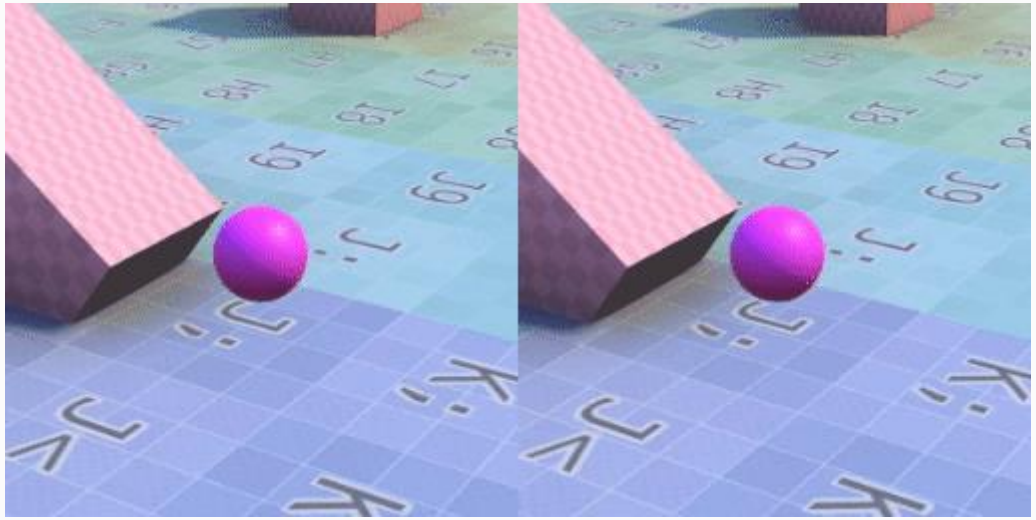
Interpolação Linear

- Para solucionar o problema de *sttutering* é necessário então realizar uma interpolação linear entre a posição obtida anteriormente pelo *update()* e a atual, criando uma suavização do movimento.
- Essa interpolação é feita levando em consideração quanto tempo falta para a próxima chamada do *update*.



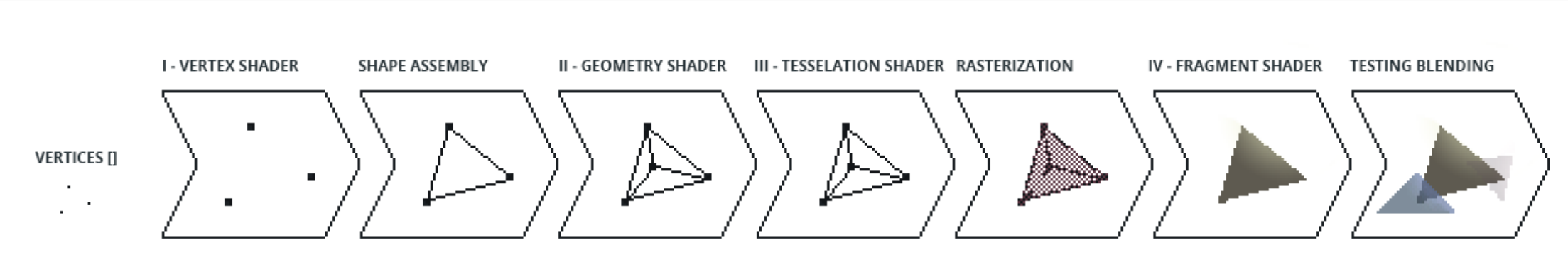


Demonstração visual



Renderização

OpenGL Pipeline



Shaders

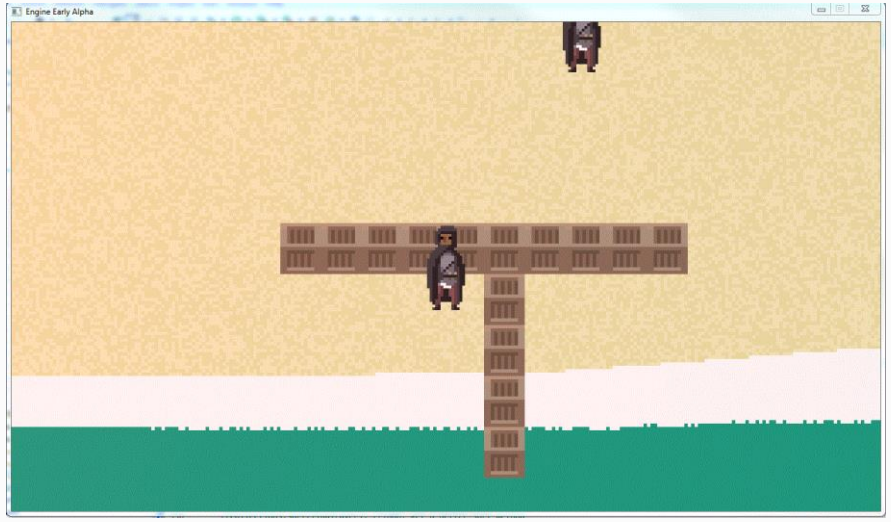
- Shaders são os pequenos programas alocados na GPU.
- Eles processam dados e os transformam em informações gráficas na tela.

Iluminação

- Existem diversos modelos de iluminação adotados em jogos.
- O mais comum deles é o **modelo de Phong**.
- Consiste em 3 componentes: luz ambiente, luz difusa e luz especular.

Iluminação **ambiente**

- A luz ambiente representa as fontes de luz mais distantes que estão quase sempre presentes no mundo real como, por exemplo, o sol.
- Ela garante a visibilidade mínima do ambiente simulando, em geral, os ciclos de dia e noite.



Iluminação **difusa**

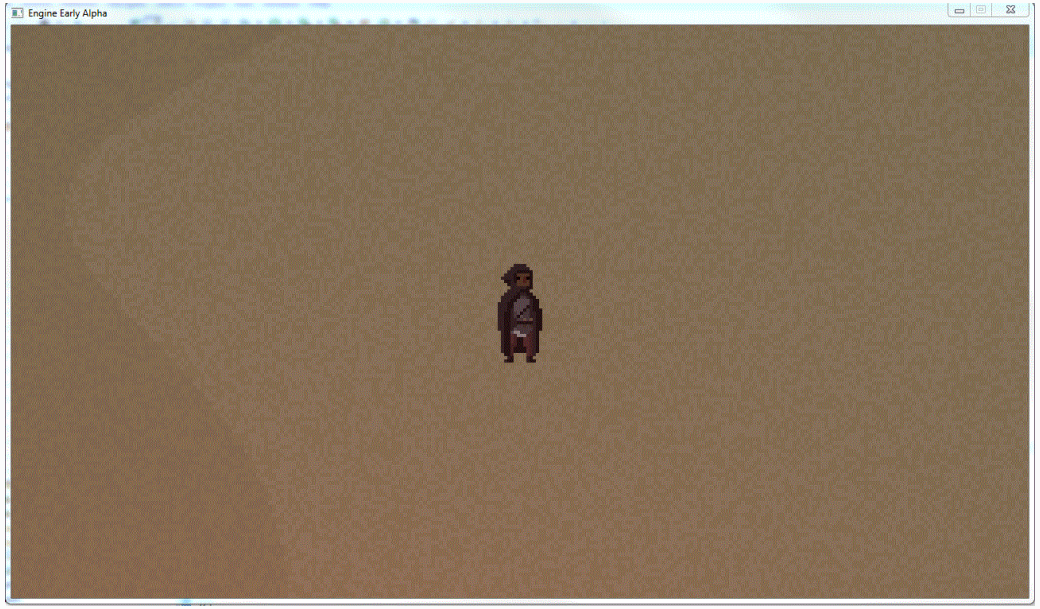
- A luz difusa simula o impacto direcional que uma fonte de luz tem sobre um objeto qualquer. Quanto mais perto o objeto está dessa fonte, mais iluminado ele será.

Iluminação **especular**

- A luz especular é o ponto brilhante que aparece em objetos lisos e reluzentes, ajudando numa melhor percepção do espaço 3D e da textura do objeto que a reflete. A luz especular é mais intensa e presente numa superfície perfeitamente lisa de, por exemplo, um espelho.

Atenuação

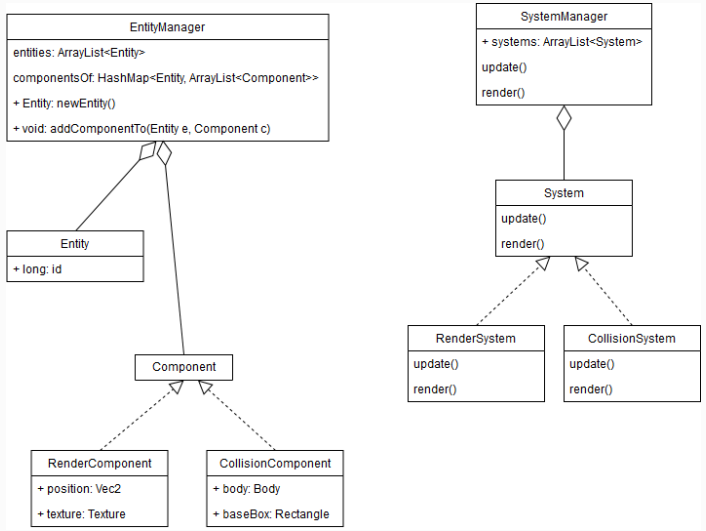
- A intensidade da luz diminui quanto maior a distância de um objeto, começando intensa e rapidamente desvanecendo.
- Para aplicar este efeito usa-se a fórmula: $F = \frac{1}{(K_c + K_l * d + K_q * d^2)}$



Estrutura lógica

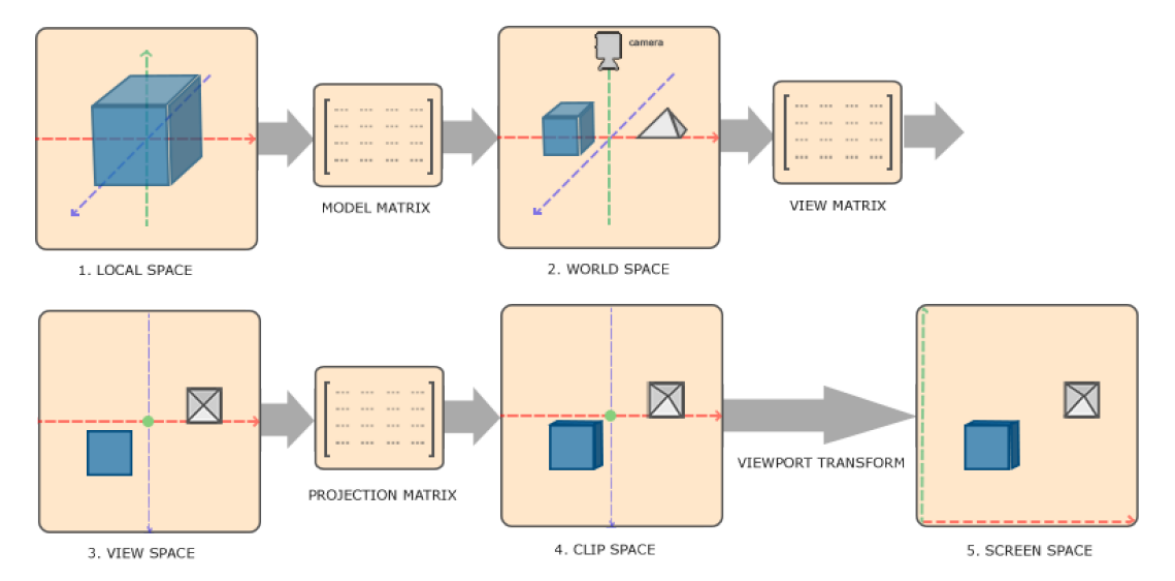
Entity Based

- A estrutura baseada em entidades opera de maneira similar a um banco de dados. Ela utiliza o conceito de composição.
- Ela desacopla os dados das funções. Dessa forma os componentes possuem apenas atributos e um sistema a parte é responsável por processá-los.



Sistema de coordenadas

- O sistema de coordenadas corresponde a todo o processo pelo qual as coordenadas de um objeto passam até serem convertidas em espaço 2D da tela.



Gerência de recursos

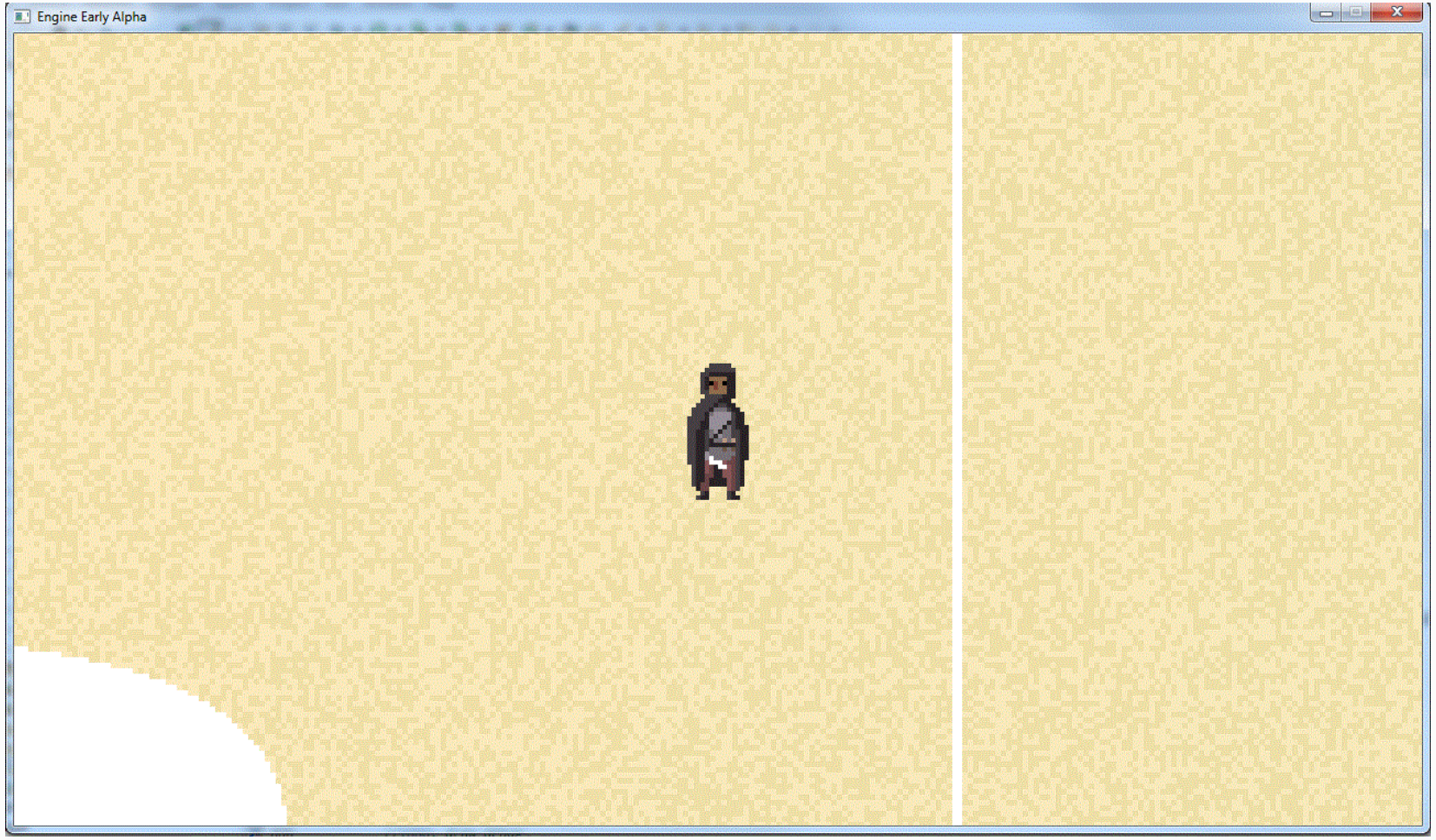
Chunks

- São pequenas frações do mapa onde estão contidos as entidades, terreno e outras informações daquele pedaço.
- Esses pedaços são objetos serializáveis para leitura e escrita em disco.
- Um conjunto de chunks salvo em disco representa o estado do jogo (game save).

Sistema de chunks

- Responsável por gerenciar as chunks e determinar quem será carregado ou descarregado.
- Funciona como uma matriz 3x3 onde haverá, no máximo, 4 chunks sendo renderizadas em tela por vez. Sendo estas aquelas que intersectam a câmera.

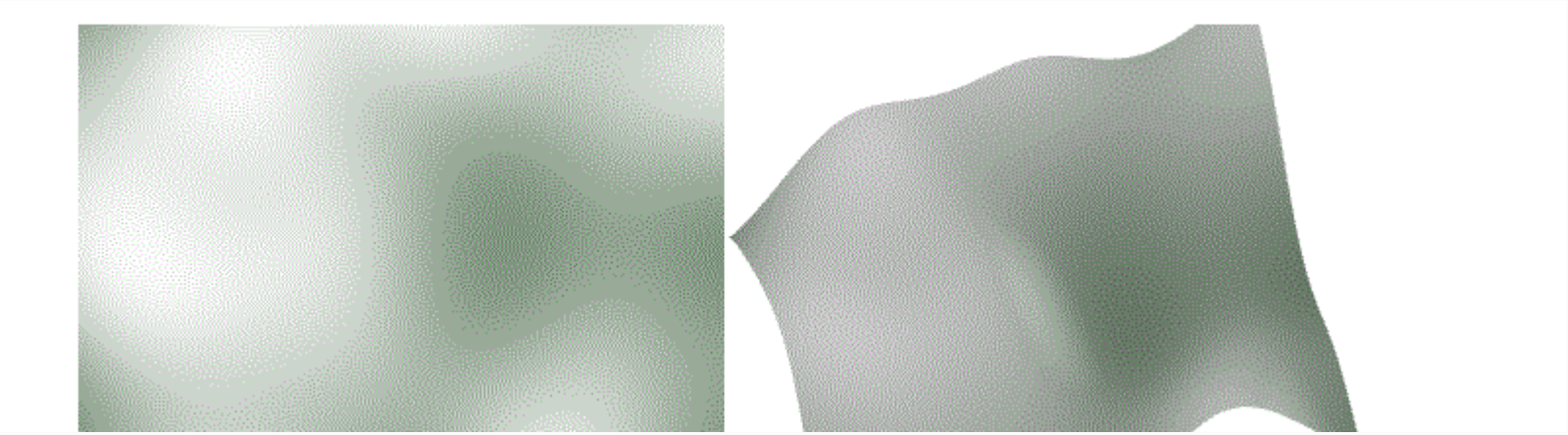
Demonstração visual



Geração de ruído e mapas

Perlin Noise

- Algoritmo para geração de um ruído menos “*machine like*”.
- Utiliza vetores gradiente para sua geração.



Gerando mapas

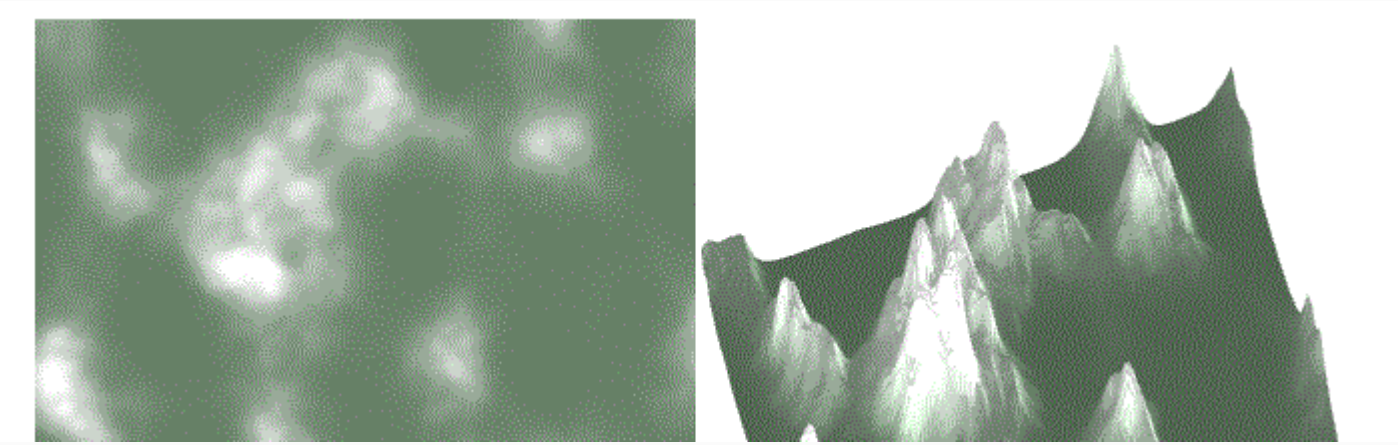
- Os valores gerados pelo perlin noise recebem significado através de regras condicionais e matemáticas.
- Para a geração do mapa na engine a fórmula padrão utilizada é:

$$d = 2 * max(abs(x - center_x), abs(y - center_y))$$

$$noise = noise + a - b * d^c$$

Atribuindo significado

- Com os valores armazenados em *noise* atribui-se uma cor baseada na interpretação de que, quanto mais próximo de 1, mais alto é aquele terreno e, quanto mais próximo de -1, mais baixo ele é, tornando-se água.



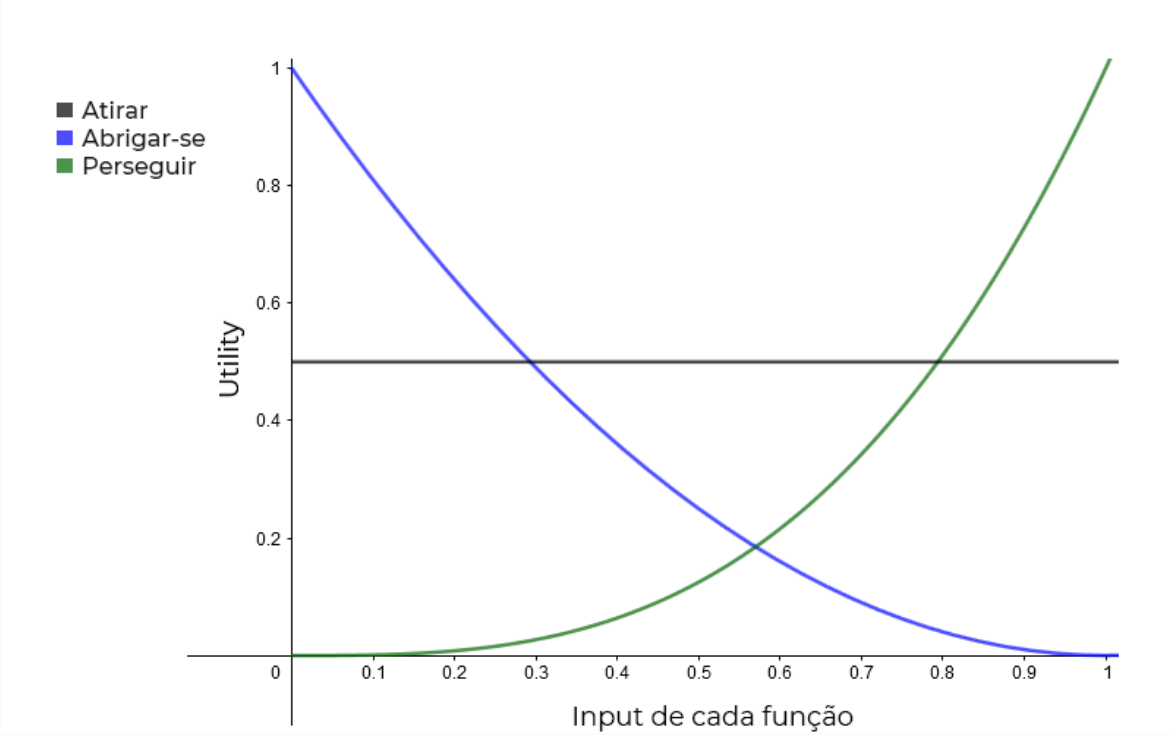
Inteligência Artificial

Utility AI

- Utility AI é o desejo de uma entidade em realizar determinada ação dado certo valor do utility, calculado com base no contexto atual.
- Para as três ações atirar, abrigar-se e perseguir tem-se o seguinte exemplo de funções:
    - Atirar: $return\ 0.5$
    - Abrigar-se: $return\ (vida - 1)^2$
    - Perseguir: $return\ distancia^3$

Utility AI

- Dessa forma o gráfico das três funções fica como se segue:

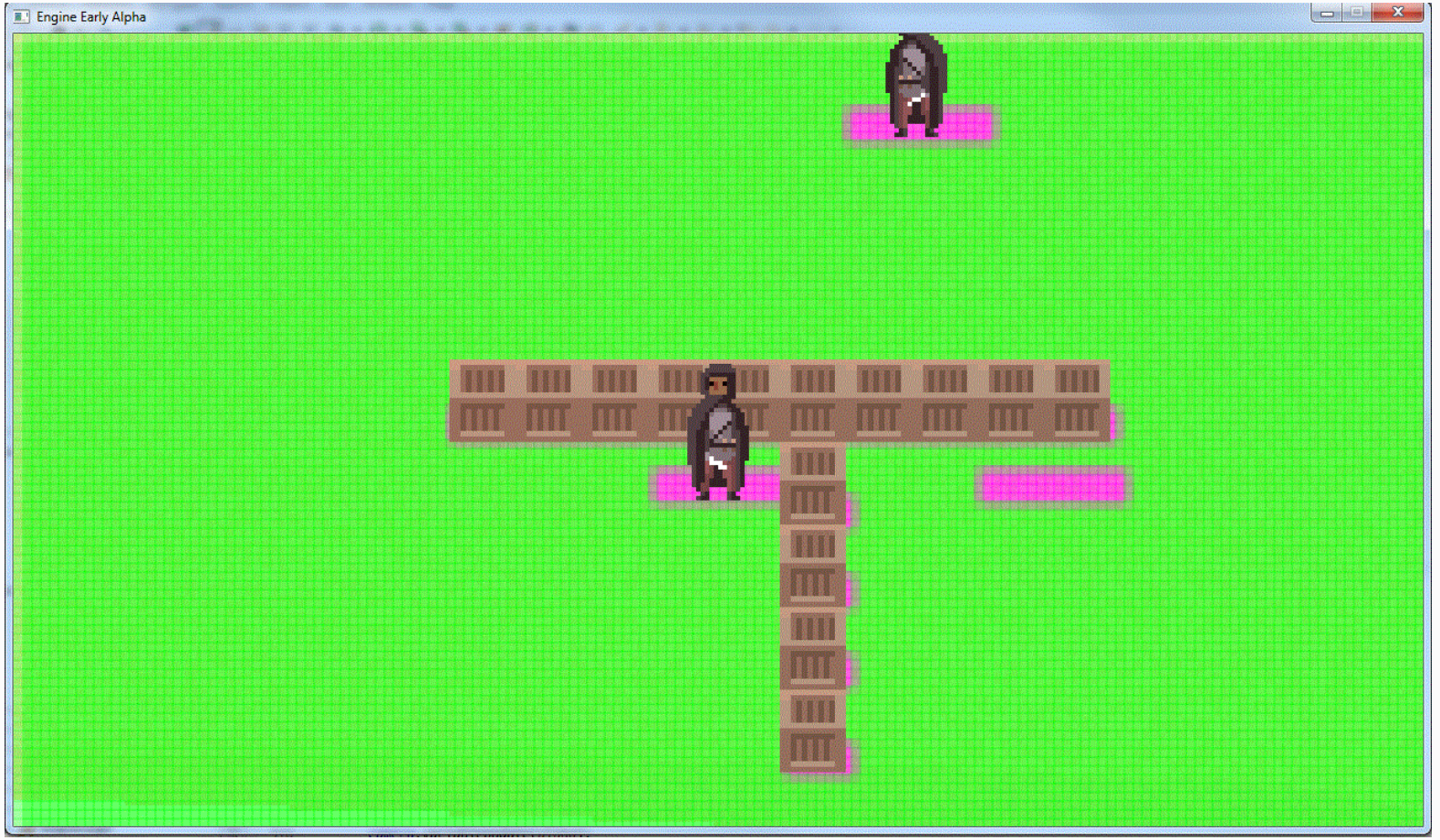


Pathfinding

- O algoritmo de pathfinding utilizado por padrão na engine é o **A***.
- O sistema subdivide o espaço visível em um grid de pequenos retângulos e então mapeia a base box de cada entidade para essas células.

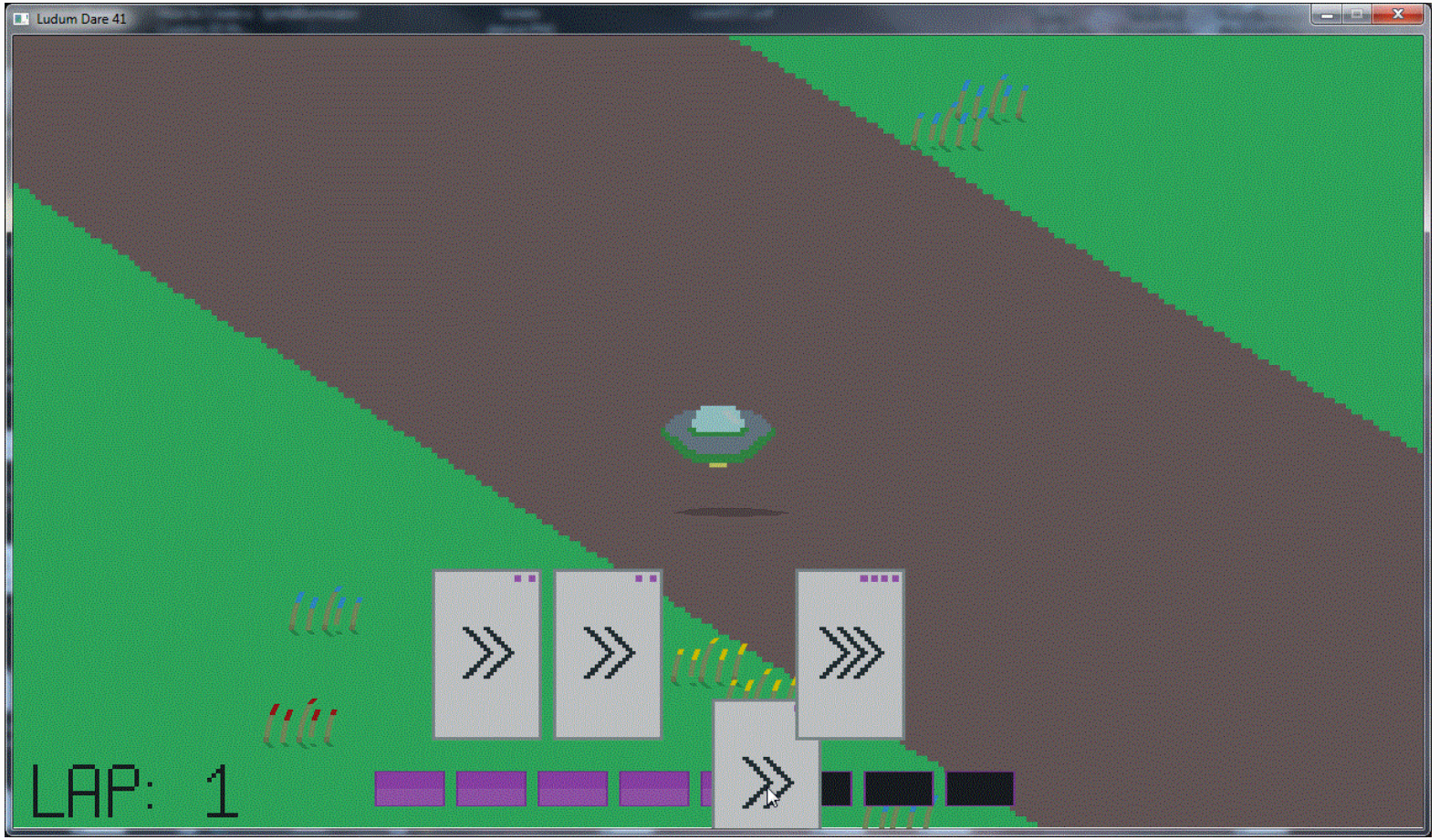
Demonstração visual sem grid

Demonstração visual com grid



Resultados e demonstrações

Ludum Dare 41



Demonstração

Demonstração

Referências

- Imagem slides 9 e 10 – Kinematicsoup, Timesteps and achieving smooth motion in unity.
- Imagem slide 24 - Learn OpenGL, Joey de Vries.
- Imagem slide 32 e 34 – RedBlobGames, Making maps with noise functions.
- Citação slide 16 - PHONG, B. T. Illumination for computer generated pictures. *Communications of theACM*, ACM, v. 18, n. 6, p. 311–317, 1975.
- Citação slide 32 – PERLIN, K. An image synthesizer. 1985.

Muito Obrigado!