Licenciatura em Engenharia Informática

Universidade do Algarve

Programação Orientada a Objetos

2023/2024

Projeto OOPS

Grupo Nº10

Jorge Silva Nº 67427

Paulo Martins Nº 71278

Vasile Karpa Nº 74872

Objetivos do Projeto

Este projeto visa desenvolver uma versão adaptada do jogo "Snake" para ilustrar conceitos da disciplina de programação orientada a objetos. O jogo permitirá a configuração e interação com vários elementos, incluindo a arena de jogo, a snake, a comida, os top N resultados, os controlos de jogo e os obstáculos que são gerados automaticamente. Nesta segunda fase, apresentamos os aspetos que consideramos mais relevantes para o desenvolvimento que realizamos até o momento. Acreditamos que ainda há espaço para melhorias neste projeto, que abordaremos mais a frente.

Aspetos Relevantes do Projeto - Rasterização

1. **Arena de Jogo**

   - A arena de jogo é dimensionada automaticamente para o múltiplo superior mais próximo do input do utilizador de acordo com o tamanho da cabeça da cobra introduzido, de forma a garantirmos que as proporções se encontram corretas e permitem uma interação entre objetos correta. Aspeto este integrado na forma de uma cabeça de tamanho 2, gera um quadrado de lado 2, e por sua vez cada lado ocupa 3 pontos, logo a arena será sempre o múltiplo maior de 3 logo a seguir ao input do utilizador.

     Ex: Head = 2, Arena 20 x 20, obtemos uma Arena 21 x 21

   - As dimensões da arena criam paredes intransponíveis que delimitam o movimento da snake e delimitam o espaço onde pode ser gerada comida.

   - É permitido criar qualquer tipo de arena, quadrangular ou retangular! A menor arena possível é de 10x10 e não existe um limite superior para a criação da arena. Porém, apesar de deixarmos a opção em aberto, cremos ser mais “user friendly” o utilizador usar uma arena de dimensões adequadas ao seu ecrã. Este aspeto ainda é motivo de debate entre os elementos do grupo.

   - A arena é renderizada novamente cada vez que existe uma alteração no estado do jogo (cobra move-se 1x numa direção por exemplo).

2. **Snake**

   - Composta por quadrados de dimensão configurável (i x i), com um tamanho mínimo de 1, seja isto, um quadrado de tamanho de lado 1, será a menor cabeça que podemos criar.

   - Relativamente ao limite superior da cabeça, só podemos criar uma snake com cabeça de lado inferior a 1/5 do tamanho da arena, seja para a sua largura ou altura.

- A snake cresce adicionando à sua cauda um segmento do mesmo tamanho da cabeça sempre que consome comida.
- A snake consome a comida gerada, que será sempre até ao tamanho máximo da sua cabeça.
- A direção inicial e o movimento da snake é definido pelo jogador.
- A snake move-se uma unidade de articulação por vez, que corresponde ao tamanho de sua cabeça. Na mudança de direção, a cauda segue o trajeto da cabeça.
- A snake não pode comer a sua cauda, e não pode alterar a sua direção para uma direção oposta, ou seja, se a direção for direita, não pode mudar para a esquerda imediatamente.

3. Comida

- A comida é removida da arena assim que consumida pela snake (ou seja, a cabeça da snake sobrepõe totalmente a comida) é gerada outra comida. Apenas é apresentada uma comida de cada vez.
- A posição da comida na arena é aleatória, mas não conseguimos garantir que não haja sobreposição com obstáculos depois de eles rodarem ou que a comida não seja gerada numa posição onde é impossível comer devido a ter obstáculos à sua frente ou causar "becos" onde após comer é garantido o 'gameover'.
- Se não for possível adicionar nova comida (é assumido que a cobra já ocupa o tamanho máximo tendo em conta os obstáculos), o jogo termina e a pontuação máxima é atribuída ao jogador, pontuação esta Integer.MAX_VALUE.
- A pontuação de cada comida é calculada através do número de pontos inteiros que a comida contém independetemente do modo de jogo (completo ou contorno), ou seja, se for um quadrado 1x1, vale 4 pontos, pois um quadrado 1x1 contem 4 pontos inteiros.
- A comida gerada para tamanhos de cabeça inferiores a 2 é sempre um quadrado, por questões de representação, em que de facto concordamos entre grupo que é uma representação mais aceitável um quadrado de lado 1 que um simples ponto, o que obrigaria a termos uma circunferência de raio 0, que não tem qualquer sentido.
- A comida gerada para tamanhos de cabeça superiores ou iguais a 2 podem ser quadrados ou circunferências de tamanho variável, e são sempre

posicionados relativamente ao canto superior esquerdo. Isto garante que é sempre possível a cobra comer a comida.

4. **Obstáculos**

- Os obstáculos podem ser triângulos, quadrados e retângulos. Estes são gerados automaticamente, posicionados automaticamente e rodados no seu centroide ou num dos seus pontos de contorno.
- A quantidade de obstáculos é dada pela seguinte fórmula *Math.min(width, height) / (Math.pow(10, String.valueOf(Math.min(width, height)).length() - 1)* que basicamente é o primeiro digito da dimensão mais pequena da arena (800x600, equivale a 6 obstáculos) e assim é gerada uma quantidade de obstáculos adequado ao tamanho da arena.
- Os obstáculos podem exceder o limite da arena, e com isto pretendemos que seja possível uma arena WxH possa ter menos obstáculos que aqueles que seriam de facto o número ideal de obstáculos, favorecendo o jogador. Claro que dependerá da sua sorte no momento da criação dos obstáculos. Pode acontecer também a situação de um obstáculo estar parcialmente na rena, e não considerámos isto um inconveniente para o nosso jogo.
- O ângulo de rotação dos obstáculos é aleatório entre [0,90,180,270]
- A colocação dos obstáculos verifica a posição da cabeça da cobra, evitando um 'gameover' logo á partida. No entanto nada impede de nascer ao lado de um obstáculo que rode, e que a sua próxima rotação cause um 'gameover' caso o jogador, faça por azar o movimento incorreto.
- Os obstáculos podem ou não rodar. A rodarem, rodam sobre o seu centroide ou sobre um dos pontos da sua borda, pelo facto de tornar o jogo visualmente mais limpo.

5. **Pontuação**

- O jogo mantém todos os resultados num ficheiro chamado resultados.txt, de onde lemos os N resultados que pretendemos apresentar, sendo N solicitado ao jogador que perdeu ou ganhou. Os recordes são apresentados do maior para o menor.
- Os pontos ingame são calculados contando o número de pontos inteiros que contem uma comida, como já mencionado anteriormente.
  (*currentScore += food.getAllIntPoints().length;*)
- Quando o jogador pressiona a tecla 'Q', sai imediatamente do jogo sem guardar os pontos. Decidimos realizar assim para já, porém assumimos que quando o 'Q' é utilizado, será quando a arena renderizada não é favorável ao

jogador, ou apenas em ambiente de teste, utilizamos ‘Q’ para sair imediatamente do jogo sem preencher mais a tabela de resultados com “lixo”.

- Quando o total de pontos feitos é zero, não guardamos esse recorde, dado que não faria qualquer sentido.

Aspetos Relevantes do Projeto - Gráfico

1. **Arena de Jogo**
    - A arena é adaptada ao input do utilizador da mesma forma que no modo de rasterização.
    - Continuamos a ter uma arena de paredes intransponíveis
    - A arena é renderizada continuamente ao contrário do modo de rasterização.
    - Continua a ser permitido gerar uma arena de qualquer tamanho, porém é sugerido uma arena 750x750.
2. **Snake**
    - A cobra é renderizada graficamente da mesma forma que era no modo rasterização, através de quadrados e funciona da mesma forma que no modo rasterização, porém agora não é apresentada passo a passo.
    - A velocidade da cobra é escolhida pelo utilizador. A alteração que é de facto realizada é no timer de jogo, que corre mais rápido ou mais lento. Portanto na realidade, alteramos a velocidade de todo o jogo.
    - Todas as regras implementadas anteriormente continuam válidas para a cobra, isto é, não pode comer a sua própria cauda, não pode mudar para uma direção oposta diretamente. Porém existe um detalhe neste último ponto, se a mudança for feita “WA” ou “SD” por exemplo, rapidamente, ela muda para o sentido oposto, o que não pretendíamos inicialmente, mas como é testa a destreza do jogador, optamos por permitir.
    - É sugerida uma cobra de tamanho de cabeça “50”, para enquadrarmos o seu tamanho na arena correspondente sugerida. É permitido qualquer tamanho dentro dos limites impostos no modo de rasterização.
3. **Comida**
    - A comida é representada da mesma forma que anteriormente, porém como utiliza mais pontos, tem um aspeto gráfico mais interessante. É apresentado da mesma forma, quadrados e círculos.
    - Relativamente aos pontos que cada comida vale, estes são mais expressivos dado o seu conteúdo conter mais pontos, porém as regras são exatamente as mesmas que para o modo de rasterização.

- As restantes regras anteriores mantêm-se aplicáveis neste modo, porém sem o problema de termos obstáculos de tamanho “1”, que seriam impraticáveis neste modo.

4. **Obstáculos**
    - Todas as regras dos obstáculos presentes no modo de rasterização mantêm-se neste modo gráfico, da mesma forma.
5. **Pontuação**
    - Os pontos são apresentados numa janela ao lado do gamescreen, demonstrando em tempo real os pontos a cada momento.
    - Os pontos neste modo têm maior peso, dado que são renderizados mais pontos, porém a lógica é a mesma, é calculado o número total de pontos da comida.
    - A qualquer momento o jogador pode consultar a tabela de pontos.
6. **Inteligência Artificial**
    - Conseguimos implementar uma Inteligência Artificial muito rudimentar que verifica se a cobra não sai da arena, não come a cauda e claro se dirige para comer a comida. Infelizmente por vezes a cobra pode-se encurralar a si própria e acaba por comer a sua cauda gerando um ‘gameover’, foi algo que já não conseguimos corrigir a tempo.

Padrões do Projeto

1. **Inputs do utilizador**
    - Modo de Rasterização (0) ou gráfico (1): int
    - Modo de Rasterização contorno (0) ou completo (1): int
    - Contem obstáculos (0) ou não (1): int
    - Tamanho da arena WxH: int x int
    - Tamanho da cabeça da cobra: int
    - Nome do jogador: String
    - N recordes: int
    - Controlos: ‘WASD’ para mover a cobra (String)
    - Controlos Gerais Rasterização: ‘Q’ para sair (String)
    - Controlos Gerais Gráfico: ‘Esc’ para sair da aplicação sem guardar, ‘Q’ para sair para o menu inicial sem guardar, ‘P’ para pausar/continuar
    - Exit Jogo: ‘Esq’ (String)

- Optamos pela interação com botões nos menus gráficos serem através do Rato. E não utilizando uma letra correspondente do Teclado como tínhamos idealizado no entregável 1.

2. Visualização do jogo

Utilizamos o modo de rasterização, que é impresso em linha de comandos, passo a passo, de forma a visualizarmos o nosso jogo iterativamente através de uma representação o mais próxima possível ao enunciado. A nossa representação de objetos, pode ser apenas dos contornos ou completa, incluindo obstáculos ou não.

Relativamente ao modo gráfico, é apresentada uma janela inicial, como estava idealizado no entregável 1, de forma a tornar o ambiente mais intuitivo ao utilizador. Neste painel inicial, podemos simplesmente jogar e definir vários aspetos relacionados com o jogo, nomeadamente o tamanho da arena, tamanho da cobra, obstáculos ou não, modo gráfico e a velocidade de jogo. Podemos ainda escolher jogar com uma Inteligência Artificial, mas sem obstáculos.

Finalmente, durante o jogo é apresentado um sidescreen que demonstra todos os dados relevantes para o jogador.

3. Placeholders

- É mostrado o número de pontos do jogador
- Apresenta a direção da Snake
- Apresenta o tamanho da cabeça da cobra
- Apresenta os controlos
- Apresenta a dimensão da arena

4. Controladores de Jogo

Os controladores de jogo, estão definidos em class própria e utilizamos as teclas:

- Key: A -> esquerda
- Key: W -> cima
- Key: S -> baixo
- Key: D -> direita
- Key: Q -> exit jogo
- Key: P -> pausa (modo gráfico)
- Key: Esc -> sai da aplicação (modo gráfico)

5. Tratamento de Erros

Dada uma implementação test-driven, serão concebidos e desenvolvidos uma série de testes para assegurar o funcionamento correto do jogo. Testes estes em todos os aspetos do jogo que consideramos relevantes.

Conclusões e Observações

Desde o início, que todos os elementos do grupo concordam em realizar uma implementação em que tornamos tudo automático ao utilizador, que apenas tem de definir alguns aspetos iniciais relevantes para o seu jogo, e a partir daí o jogo será sempre diferente, em que o nível de dificuldade nunca será o mesmo, pois nunca sabemos que obstáculos serão gerados, nem como eles vão funcionar (rodar sobre que ponto e quantos graus, ou não rodar). Isto confere uma experiência muito mais agradável ao jogador na medida que se torna um jogo mais desafiante e difícil de bater recordes.

Relativamente à arena decidimos que o utilizador deve ser livre de criar uma arena do tamanho que quiser desde que maior que 10x10.

Para os obstáculos, não existem polígonos pelo motivo de que o preenchimento dos mesmo no modo completo foi um aspeto que nos consumiu algum tempo, e sem sucesso numa renderização bem-sucedida para todos os casos que testámos. Outro aspeto que não foi bem concretizado foi gerar os pontos de um polígono aleatoriamente, para N vértices escolhidos aleatoriamente entre 5 e K (K é inteiro maior que 5). Dado que existiram vários entraves que nos impossibilitaram de mantermos a estrutura que já disponhamos anteriormente para os outros obstáculos (quadrado, triangulo e retângulo), optamos por descartar os polígonos no nosso jogo.

Cremos que o facto de gerarmos tudo automaticamente é de facto uma mais-valia no nosso projeto, ao invés de, utilizando a main dada pelo docente para a resolução dos últimos problemas propostos na cadeira, para preencher um ficheiro obstaculos.txt que conteria todos os obstáculos que poderíamos utilizar, e apenas teríamos de escolher aleatoriamente entre eles, e renderizar os seus contornos.

Outra observação é o facto de a geração de comida numa posição aleatória pode por vezes puder causar um ‘gameover’ dada a sua posição não permitir que a cobra consiga prosseguir o seu trajeto. Esta situação é algo que ainda nos causa algum transtorno, porém é um problema que ainda estamos a tentar resolver. De facto, calcular todos os pontos possíveis de rotação de um obstáculo (que por sua vez pode rodar sobre diferentes pontos, e com diferentes rotações, todas calculadas aleatoriamente) e verificar a todos os momentos se a comida não é gerada nesses mesmos pontos, ainda não foi conseguido com sucesso. Observamos também que esta situação é muito mais difícil de ser concretizada no modo gráfico dado o tamanho da arena ser muito maior. A quantidade de obstáculos calculada para este modo é realizada calculando o primeiro dígito do tamanho mais pequeno da arena. Ex: 500x400 o resultado é 4 obstáculos

Em relação ao modo automático, como foi dito conseguimos implementar uma Inteligência Artificial muito rudimentar que ainda conta com alguns problemas que poderiam ser resolvidos com mais algum esforço e tempo.

Relativamente á velocidade da cobra, definimos que a variável que iremos alterar é de facto o “timer”, alterando a velocidade do jogo e não a velocidade apenas da cobra, permitindo ao jogador escolher entre várias velocidades. Com isto pretendemos atribuir vários níveis de dificuldade ao jogo.

Os menus de jogo implementados foram os menus previstos no entregável 1, porém com mais algumas características que consideramos necessárias e inclusive a alteração do layout das mesmas de forma a corresponder às necessidades do jogador.

Relativamente ao aspeto gráfico, foi algo que definimos da forma que fosse o mais percetível possível ao jogador, através das cores, vermelho (obstáculos), verde (cobra) e rosa (comida), e quando atingimos um obstáculo com a cobra ele muda para amarelo.

Finalmente, relativamente ao modo gráfico, durante o jogo, temos alguns problemas com as colisões entre cobra e obstáculos, já que estas colisões não funcionam corretamente. Porém é um problema identificado que tentamos resolver! Infelizmente sem sucesso.