Programação Modular
Trabalho Final
Visualização Interativa de Agentes Inteligentes

Daniel Diniz - 2008046090 - danieldiniz@dcc.ufmg.br
Diogo Santana - 2011054308 - diogo.marques@dcc.ufmg.br
Frederico Figueiredo - 2010054371 - fredfig@dcc.ufmg.br
Sandro Miccoli - 2009052409 - smiccoli@dcc.ufmg.br

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

17 de junho de 2015

Resumo. *Esse relatório descreve como foi implementado a Visualização Interativa de Agentes Inteligentes, proposto como tema para o Trabalho Final. A Seção 1 introduz o problema proposto e dá uma visão geral da solução implementada. Cada seção irá descrever detalhes do sistema desenvolvido, abrangendo desde o planejamento (Seção 2), as decisões de implementação (Seção 4) e os testes realizados (Seção 6). Finalmente concluímos (Seção 7) a documentação com reflexões sobre o aprendizado durante a execução do trabalho.*

1. Introdução

Durante as aulas da disciplina vimos muitos exemplos de como implementar formas geométricas dinâmicas, então decidimos aplicar isso no nosso trabalho final. O objetivo do trabalho é unir os conceitos que aprendemos sobre programação modular e padrões de projeto e aplicá-los num contexto que seja esteticamente agradável. Isso será feito utilizando apenas formas geométricas e linhas com o intuito de gerar padrões visuais interessantes. O modo como chegaremos nesse resultado será utilizando algoritmos de movimentação de inteligência artificial, usando conceitos como bug algorithm, campos de potencial e desvio de obstáculos (steering). O planejamento inicial consiste em construir agentes inteligentes que se atraem e repelem. Cada grupo de agentes terá uma atração a certo grupo e repulsão a outros. Por causa dessa característica de atração e repulsão, no momento em que vários agentes são atraídos por um único agente, é esperado que eles entrem em equilíbrio e formem padrões geométricos regulares.

Modelagem inicial do problema. Construção de agentes básicos. Implementação do bug algorithm com campos de potencial. Implementar comportamentos aleatórios para os agentes. Implementação de uma interface gráfica Escrever documentação do projeto

Possível utilizar a biblioteca controlP5 do Processing, que permite alterar valores do código em tempo de execução.

O problema proposto neste trabalho foi a especificação e implementação de uma visualização jogo de cartas a ser escolhido pela dupla. O jogo escolhido por nós foi o Pôquer Chinês (Big Two). O objetivo principal do jogo é vencer ao acabar com todas as cartas da mão recebida pelo jogador. A proposta do trabalho prático é proporcionar uma

interface simples por linha de comando que seja jogável. O programa implementado será detalhado superficialmente aqui, porém a (Seção 4) irá explicar em profundo os detalhes de implementação.

2. Planejamento

O planejamento que antecedeu à implementação ocorreu da seguinte forma: estudamos as regras do jogo existentes na internet, como são muitas variantes, tivemos que escolher uma e implementar ela. Após isso, definimos os módulos principais que o jogo deveria ter, suas classes e métodos. Com isso, geramos um diagrama UML para melhor compreensão do que iríamos desenvolver. Além disso criamos um repositório git online para que pudéssemos trabalhar em paralelo no mesmo código e caminhar rápido com o trabalho.

3. Regras

O Pôquer Chinês [BigTwo] é um jogo de cartas cujo objetivo é ser o primeiro a desfazer de todas as suas cartas. É por vezes chamada de pôquer chinês devido à sua origem chinesa e uso de mãos de pôquer, embora seja um jogo de natureza completamente diferente. Ele é jogado tanto casualmente como por dinheiro. É geralmente jogado com dois a quatro jogadores, com o baralho inteiro sendo distribuído para todos os jogadores, ou 13 cartas para cada jogador. Após as cartas terem sido distribuídas, no geral, o jogador com a menor carta, o 3 de ouros, joga primeiro, com o jogo prosseguindo em uma direção horária. Cartas podem ser colocadas na mesa como únicas, duplas, trios ou em grupo de cinco, utilizando as regras do pôquer, inclusive na primeira jogada feita por quem tem o 3 de ouros. Depois do início do jogo dado pela jogada do primeiro jogador, o jogador a seguir precisa colocar um mesmo número de cartas que sejam maiores do que aqueles colocados pelo jogador anterior. Caso um jogador não possa ou não queira colocar uma carta ou combinação, ele passa seu turno. Caso todos os oponentes decidam passar, então o jogador remanescente possui o direito de colocar qualquer combinação permitida. O naipe das cartas é utilizado como critério de desempate, sendo a ordem do menor para o maior: ouros, paus, copas e espadas.

3.1. Tipos de combinações de jogo

Essa subseção irá listar todas as possíveis combinações de jogo.

Uma carta

Qualquer carta do baralho, ordenadas pelo valor da carta (sendo 3 o menor e 2 o maior), com o naipe sendo o critério de desempate. Por exemplo, o 2 de espadas é maior do que qualquer outra carta do jogo, o A de espadas é maior do que o A de ouros, esta sendo maior do que um K de espadas.

Par

Quaisquer duas cartas iguais, com o maior naipe das duplas sendo utilizado como critério de desempate. Um par de K de espadas e K de ouros é melhor que um par de K de copas e K de paus.

Trinca

Três cartas de mesmo valor, o desempate é feito através do maior valor.

Cinco cartas

Existem cinco mãos válidas, ordenadas a seguir do menor para o maior:

Sequência

Cinco cartas em sequência, não do mesmo naipe. Caso empate ganha aquele com a maior sequência: 10-J-Q-K-A >... >2-3-4-5-6 >A-2-3-4-5, ou seja, o rank da maior carta da sequência determina qual é a maior, o A e o 2 nas sequências baixas não influenciam no valor da sequência;

Flush

Cinco cartas do mesmo naipe, não em sequência. O desempate é determinado primeiro pela carta de maior valor, e então pelo naipe mais alto;

Full House

Uma trinca e um par, a trinca mais alta é utilizada como critério de desempate;

Quadra + 1

Quatro cartas do mesmo valor mais uma carta qualquer. Observe que, ao contrário do pôquer, uma quadra não pode ser utilizada por sí só. O desempate é determinado pelo valor de uma das cartas da quadra.

Sequência de mesmo naipe

Uma sequência e um flush ao mesmo tempo. O desempate é determinado primeiro pela maior carta, e então pelo maior naipe.

Sequência real

O mesmo que a sequência de mesmo naipe, mas limitado a 10, J, Q, K e A. Esta é a maior combinação possível. Apenas uma sequência real com um naipe maior pode vencer outra sequência real. A sequência real de espadas é a combinação de cartas mais poderosa do jogo.

4. Implementação

Para a implementação dessa simulação utilizamos duas biblioteca de classes Java, **Processing** e **ControlP5**. O conjunto de funções presentes nas classes do Processing facilitam o desenvolvimento de aplicativos orientados a arte visual e contextos gráficos em geral, enquanto as presentes em ControlP5 nada mais que auxiliam a implementação de controladores para elementos integrados às bibliotecas do Processing.

Foram utilizados neste sistema os padrões de projeto *Singleton, Decorator e Observer*, de modo a garantir uma melhor modularização do código. O primeiro foi utilizado para assegurar uma única instância da classe responsável pela apresentação gráfica do sistema, bem como facilitar sua recuperação e referenciamento por outras classes do aplicativo. O mesmo padrão foi utilizado para tratar a instanciação da classe ControlP5Panel, garantindo que quaisquer alterações feitas no painel ou qualquer informação que seja buscada por outras classes partirá de uma única instância unificada deste, tornando-o escalável, dinamicamente modificável, estável e persistente.

Neste painel pode se constatar, também, a aplicação do padrão *Observer*. Durante a instanciação do Singleton, é criado um elemento nominado *CallbackListener*, que escuta por interações do tipo *ACTION_PRESSED* nos elementos do painel de controle, capturados apenas na instância gráfica principal do aplicativo. Essas interações são então convertidas na forma de eventos e enviados de volta para tratamento na classe do **P5ControlPanel**. Tudo isso a fim de garantir que as devidas funções permaneçam encapsuladas de acordo com os princípios de modularidade.

O padrão Decorator por sua vez foi implementado objetivando limitar o número de alterações necessárias na forma básica - *circle* - e permitir que novos incrementos de formas e comportamentos sejam feitos respeitando os conceitos fundamentais de Orientação a Objetos, ou seja, grande escalabilidade com o mínimo de alteração nas classes mais abstratas e estáveis.

A interface (Shape) representa uma forma geométrica que pode ser 'decorada' com uma infinidade de elementos diferentes. Nesse projeto implementamos apenas diferentes tipos de círculos, tratados para todas as finalidades como comportamentos que decoram os círculos. A simulação conta com diversas instâncias da forma *circle*, que são independentemente decoradas com o devido comportamento de acordo com a interação realizada pelo usuário através do painel de controle. Tais forças extendem a classe DecoratorShape, adicionando funcionalidade e efetivamente implementando à interface Shape.

Tudo isso é então colocado em execução sincrôna na classe Main, que sobrescreve funcionalidades implementadas na biblioteca processing para criar uma thread que monta a simulação com as configurações iniciais e mantém todos os elementos em constante atualização e movimento.

Segue o diagrama de classes simplificado do sistema:

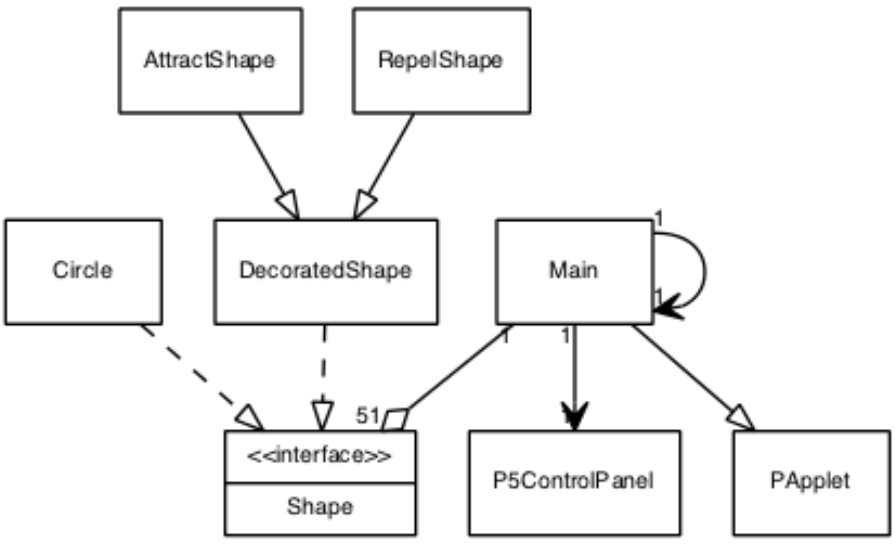

Figura 1. Diagrama de classes simplificado

5. Funcionalidades

O programa inicia com 6 círculos em posições, direções e velocidades aleatórias, além de um ponto no centro da tela representando o centro de massa de atraçãorepulsão. Outros comportamentos podem ser adicionados a partir das seguintes teclas ou botões:

Botão Vectors
Mostra/esconde o vetor de direção do círculo.

Botão Connect
Mostra/esconde as linhas que ligam os centros de todos os círculos.

Botão Attract
Atrai todos os círculos para o centro de massa.

Botão Repel
Aumenta a repulsão entre os círculos.

Botão Reset
Desfaz todos os comportamentos adicionados, retornando a execução ao estado inicial.

Botão Trail
Diminui a frequência de atualização do background da tela, ou seja, diminui a taxa na qual o fundo da tela é redesenhado, mantendo desenhos antigos na tela.

Botão Size
Aumenta/reduz o tamanho dos círculos.

Botão RepelIntensity
Aumenta/reduz a distância de repulsão entre os círculos.

Barras horizontais
Alteram a cor das linhas criadas pelo botão Connect, sendo as 3 primeiras barras referentes às cores RGB e a última referente à opacidade.

Botão Add
Adiciona um novo círculo, sendo o máximo possível igual a 30.

Botão Remove
Remove o último círculo adicionado contanto que o número de círculos na tela seja maior que 5.

Tecla Espaço
Adiciona um círculo atrator, sendo o máximo possível igual a 10. Não é afetado pelas demais funcionalidades, exceto pelo *Trail*.

Tecla Enter
Adiciona um círculo repelidor, sendo o máximo possível igual a 10. Também é afetado somente pelo *Trail*.

Tecla Tab

Mostra/esconde o menu de opções.

Tecla Shift + P

Tira um screenshot da tela e salva na pasta *prints*

6. Testes

O desenvolvimento deste aplicativo passou por diversas fases até chegar ao estado de arte em que se encontra. Por se tratar também de um conceito que pode ser considerado bastante abstrato por alguns, não havia, inicialmente, uma ideia fixa de como seria o estado final do programa. Assim, seguem algumas imagens das primeiras versões e ideias que tivemos, que podem ser consideradas também como testes para descobrirmos os potenciais que poderiamos atingir, além de servir como motivação para a evolução incremental da ideia.

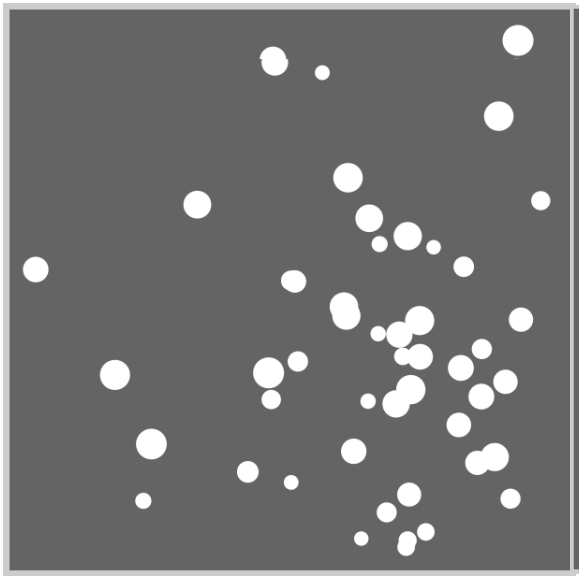

Figura 2. v0.1

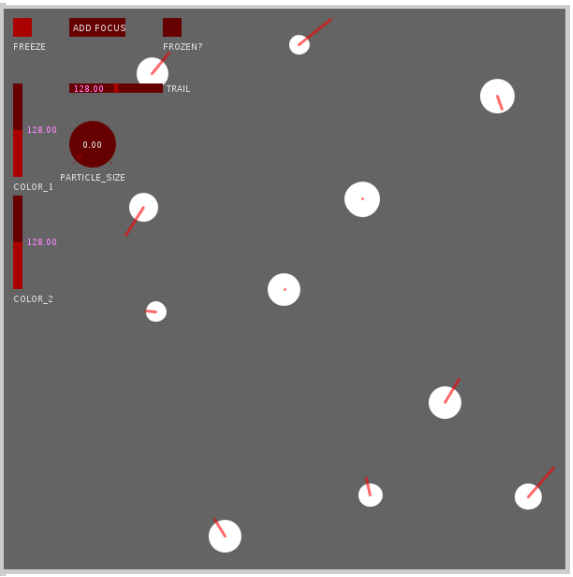

Figura 3. v0.5

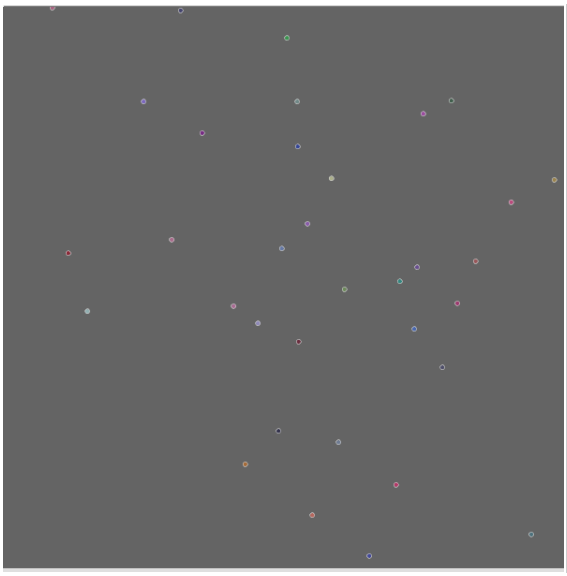

Figura 4. v0.8

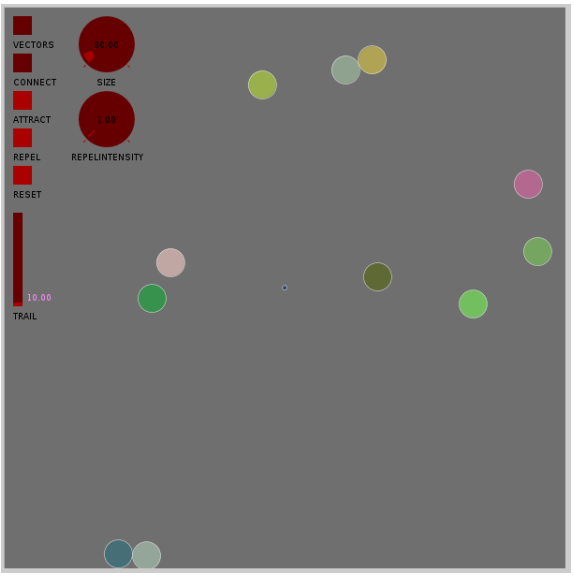

Figura 5. v1.0

7. Conclusão

A partir deste trabalho foi possível aprimorar nossos conhecimentos em programação modular pela aplicação conceitos de padrões de projeto, interface gráfica, herança e encapsulamento de dados. Além do mais, o projeto também proporcionou uma experiência de desenvolvimento em equipe e da modelagem em conjunto de um problema, aproximando-nos à realidade de desenvolvimento do mercado.

Apesar das dificuldades de implementação do padrão de projeto Decoratir, os resultados mostraram-se satisfatórios e até mesmo melhores do que o esperado. Gerar padrões de imagens mostrou-se bastante divertido e foram gastos horas apenas experimentando as diversas combinações de funcionalidades

Referências

BigTwo. Página do wikipédia sobre o big two. `http://en.wikipedia.org/wiki/Big_Two`.

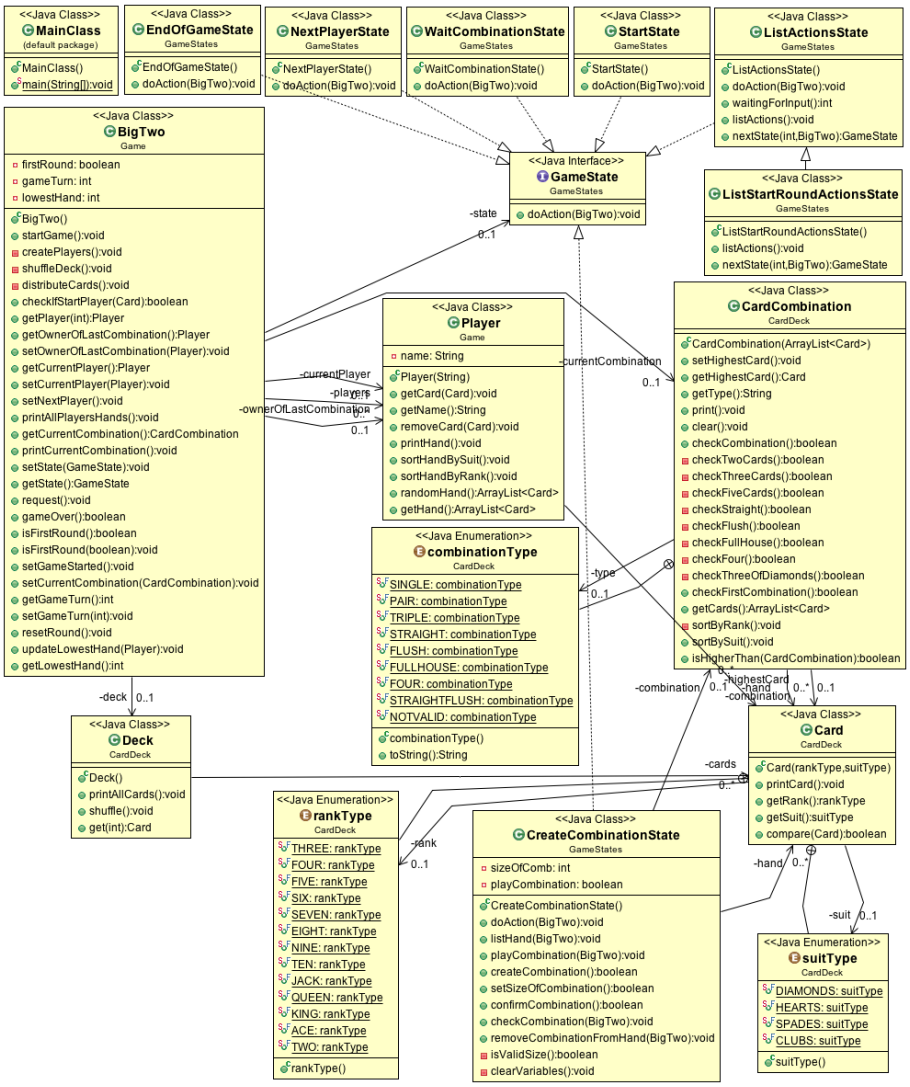

Figura 6. Diagrama de classes completo