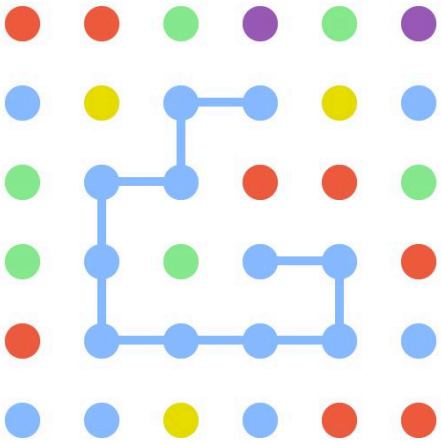
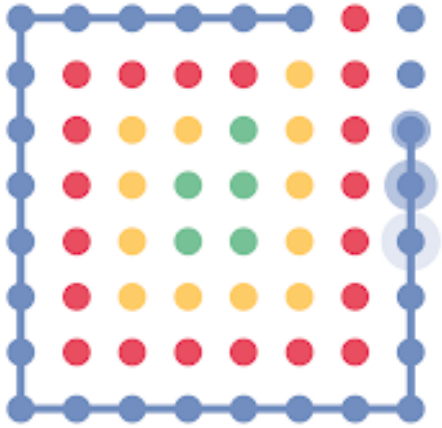
Programação 2017/2018

Mestrado em Engenharia Electrotécnica e de Computadores (MEEC)





Projeto de Programação: Avaliação Intermédia

1 Introdução

Com este projeto pretende-se que os alunos desenvolvam um jogo intitulado ISTDots, em que o objectivo é juntar pontos da mesma cor colocados num tabuleiro e assim eliminá-los. Existem várias variantes deste jogo e várias aplicações para iPhone/Android sendo uma das mais famosas a versão desenvolvida pela Betaworks. O jogo apareceu em 2013 e passado uma semana depois de ter sido lançado, já tinha sido descarregado mais de 1 milhão de vezes e era a 1ª aplicação em vários países. Para poderem perceber melhor a dinâmica do jogo podem instalar a aplicação disponível em www.dots.co.

2 Dinâmica do Jogo ISTDots

A dinâmica deste jogo funciona de acordo com as seguintes regras:

- O jogador pode ligar linhas de pontos na horizontal ou na vertical (mas nunca na diagonal) da mesma cor de forma a eliminar todos os pontos dessa linha. Pode ver um exemplo na figura em cima.
- O jogador pode criar um quadrado unindo pontos com a mesma cor e assim elimina todos os pontos dessa cor no tabuleiro do jogo. Além disso, todos os pontos dentro do quadrado (se existirem) devem ser removidos, independentemente da sua cor.
- Após cada jogada, dois ou mais pontos devem ser movidos de forma a que os outros pontos possam cair de uma forma natural. Os pontos deslizam na direção para baixo até que sejam parados por

qualquer outro ponto ou pela borda do tabuleiro. Novos pontos devem aparecer sempre de cima para baixo.

- Sempre que não for possível fazer nenhuma jogada, isto é, não existam nenhuns pontos que se possam unir, os pontos devem ser atribuídos a novas posições aleatoriamente de forma a que seja possível fazer uma nova jogada.
- No início do jogo são estabelecidos os objetivos do jogo, isto é, o número de jogadas permitidas e o número de pontos de cada cor que devem ser unidos de forma a alcançar a vitória. Estes valores devem ser atualizados ao longo do jogo.

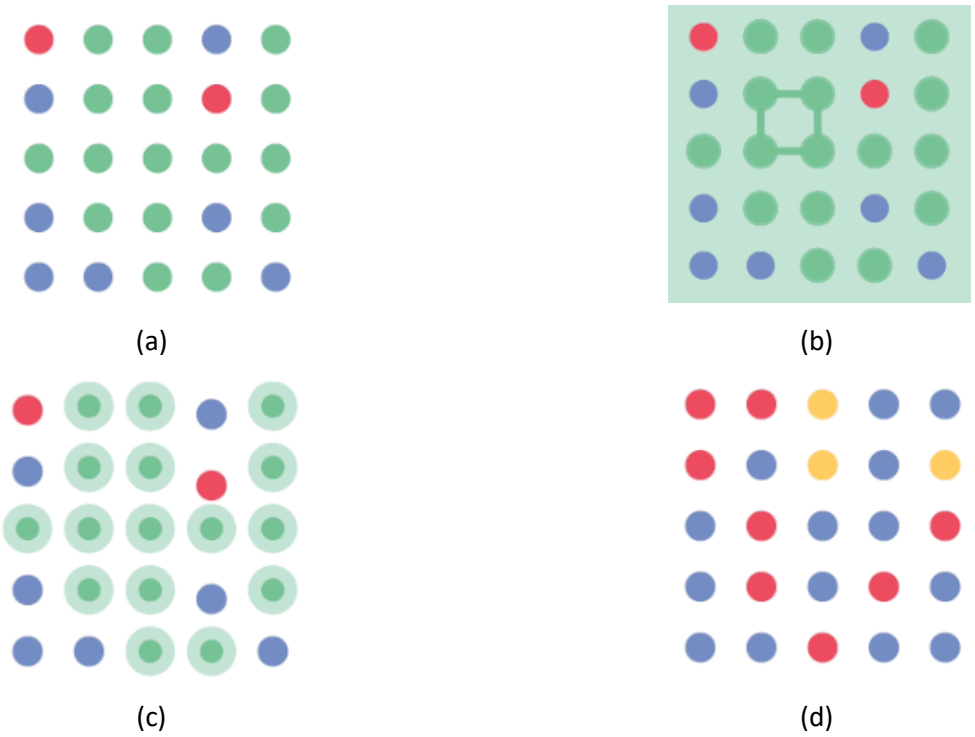


Figure 1 - Dinâmica do jogo: a) pontos de diversas cores distribuídos pelo tabuleiro, b) seleção de um conjunto de pontos a unir, c) todos os pontos assinalados são removidos e d) tabuleiro após este processo.

As derrotas e vitórias deste jogo obedecem às seguintes regras:

- **Vitória**: Sempre que o jogador conseguir alcançar os objetivos do jogo num número de jogadas inferior ao estabelecido, deve ser assinalada vitória, como ilustrado na Figura 2 b). Os objetivos do jogo são um conjunto de parâmetros lidos no início do programa (*ver Seção 4*).
- **Derrota**: Ocorre uma derrota sempre que o jogador esgotar o número de jogadas estabelecido sem que tenha alcançado os objetivos do jogo, como ilustrado na Figura 2 a).

(a) (b) (c)

Figure 2 - a) Derrota, b) Vitória, c) Sem mais movimentos disponíveis (exemplo da mensagem).

Ao longo do jogo é essencial mostrar o número de jogadas que ainda pode realizar e o número de pontos que ainda falta unir para cada uma das cores (ver Seção 5). O número de jogadas e o número de pontos (de cada cor) que ainda falta unir para obter uma vitória deve ser atualizado ao longo do jogo.

3 Funcionamento do Programa

O funcionamento do programa pode ser sumarizado da seguinte forma:

- Leitura dos parâmetros de configuração (ver Seção 4).
- Mostrar o tabuleiro como uma matriz de pontos com a dimensão especificada.
- Um jogo deve começar apenas quando o utilizador carregar na tecla **n**. A qualquer altura pode ser iniciado um novo jogo, mesmo quando o jogo atual ainda não tenha acabado, considerando que o jogador teve uma derrota.
- As cores dos pontos do tabuleiro devem ser gerados aleatoriamente bem como as cores dos pontos que aparecem. O número de cores está disponível como parâmetro de configuração. Todos os quadrados do tabuleiro devem estar preenchidos com pontos.
- Depois dos pontos distribuídos, é necessário permitir ao jogador escolher quais são os pontos que pretendem unir, utilizando para tal o rato. Assim, cada vez que o utilizador pressionar o botão esquerdo do rato pode escolher um ou mais pontos. Quando o utilizador libertar o botão do rato sinaliza que os pontos a juntar já foram escolhidos.
- Devem verificar se os pontos que foram unidos, correspondem a uma jogada válida.
- Devem remover os pontos após cada jogada, mover os pontos restantes no tabuleiro e fazer aparecer novos pontos de acordo com a dinâmica do jogo.
- Devem detetar a ocorrência de quadrados na jogada do utilizador e remover todos os pontos com essa cor (e os pontos dentro do quadrado independentemente da cor). Note-se que formas geométricas mais complexas (mas incluindo um quadrado) devem ser suportadas.
- Cada ponto removido do tabuleiro deve ser contabilizado e o número de pontos a unir dessa cor decrementado (aos objetivos do jogo).
- Devem atualizar a número de jogadas restantes e o número de pontos que falta unir após cada jogada.
- Naturalmente, não devem deixar um jogador continuar a jogar depois de este ter ganho ou perdido.

- Quando o **programa terminar**, deverá ser escrito um ficheiro de estatísticas contendo as estatísticas de todos os jogos concluídos (ver Seção 4). **Estas estatísticas devem estar guardadas num vector.**
- O jogador pode sair a qualquer momento da aplicação carregando na tecla **q**.

Naturalmente, é necessário atualizar a interface gráfica para mostrar os pontos que são unidos, sempre que este efetuar uma jogada e a atualização do tabuleiro sempre que sejam realizada uma ação válida.

Em relação ao código, é obrigatório que o tabuleiro com os pontos seja representado como uma matriz 2D de inteiros. Não podem ser usadas estruturas de dados.

Escolha **uma** das seguintes funcionalidades **avançadas** e faça a sua implementação:

- Implementação de quadrados congelados, isto é, que não podem ter nenhum ponto. Ao cair as peças devem saltar estes quadrados.
- Devem permitir ao jogador fazer *undo*, isto é, anular a última jogada feita, voltando o tabuleiro e pontuação ao estado antes de efetuar jogada. Não é necessário realizar mais do que um nível de *undo*. A tecla **u** deve ser usada para esta funcionalidade.

Não deverão implementar estas funcionalidades avançadas sem terem as restantes funcionalidades a funcionar corretamente, uma vez que poderão ser penalizados caso o façam. Caso estas funcionalidades sejam implementadas, receberá uma bonificação (a definir) na nota final.

4 Leitura dos Parâmetros de Jogo e Escrita de Ficheiros

O funcionamento do jogo deve ser ditado por um conjunto de parâmetros que devem ser introduzidos pelo utilizador (através do teclado) antes de começar qualquer jogo:

- **Tamanho do tabuleiro:** Devem suportar tabuleiros de qualquer tamanho. No entanto, não pode haver tabuleiros com menos de 5 casas e mais de 15 casas na vertical ou na horizontal.
- **Nome do jogador**: Este nome não pode ser superior a 8 letras.
- **Número de cores:** Número de cores que devem existir em simultâneo no jogo. Não pode haver mais do que 5 cores diferentes.
- **Número de pontos a alcançar:** Para cada uma das cores especificadas, o número de pontos que o utilizador deve juntar para ganhar o jogo. Não pode haver mais do que 99 pontos.
- **Número de jogadas:** Máximo número de jogadas permitidas para que o jogador obtenha uma vitória. Não pode haver mais do que 99 jogadas.

Cada parâmetro de entrada deve ser validado e caso o utilizador não tenha introduzido um parâmetro válido deve ser pedido de novo ao utilizador esse parâmetro, sinalizando a ocorrência de erro. Casos óbvios como números negativos ou letras também devem ser detetados e sinalizados.

Apenas quando o programa terminar é necessário escrever um ficheiro com o nome *stats.txt* que indique a seguinte informação para cada jogo:

- Nome do jogador.
- Número total de jogos que iniciou e o número de derrotas e vitórias.

- Para cada jogo, indicar se obteve uma vitória (V) e o número de jogadas que realizou para alcançar a vitória (ex. 30 V) ou uma derrota (D).

5 Aspecto Gráfico

A aplicação *ISTDots* deverá ter um aspeto gráfico semelhante ao que está ilustrado na Figura 3.

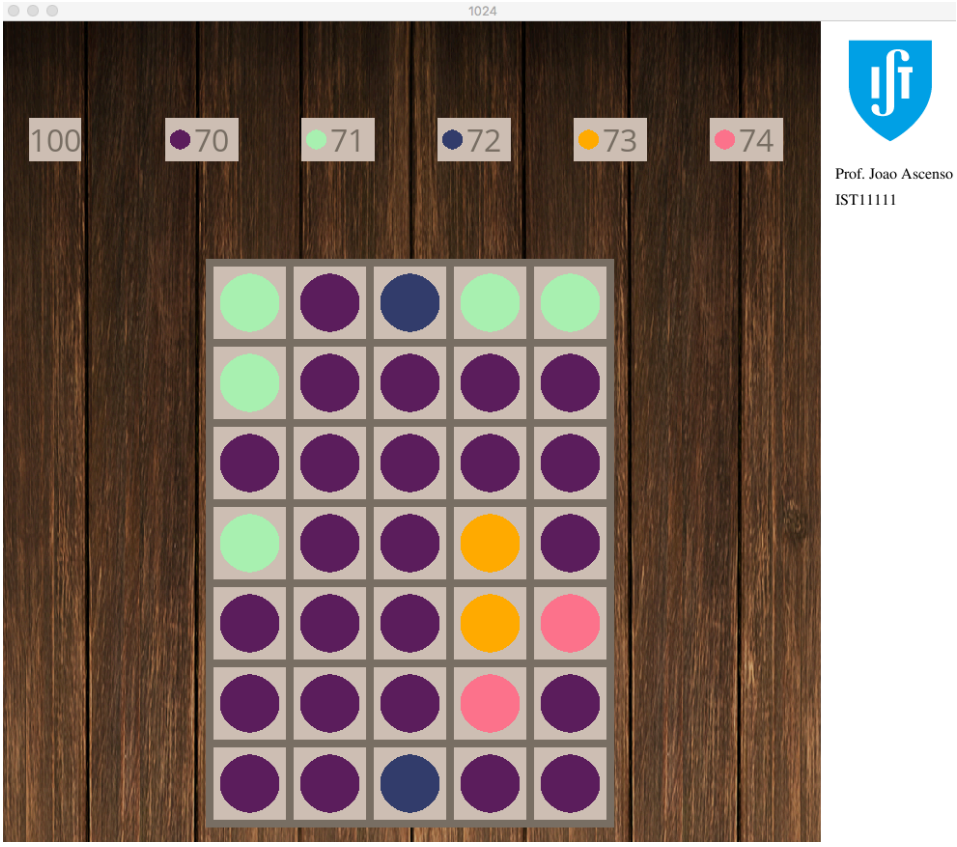


Figure 3 – Interface gráfica do jogo ISTDots. No topo da interface devem mostrar o número de jogadas, o número de pontos necessário para obter uma vitória.

A aplicação deverá mostrar o nome da aplicação (ISTDots) e o autor da aplicação (nome e número do aluno). Para realizar a parte gráfica devem utilizar o código de apoio ao projeto. No entanto, note-se que necessitam de efetuar as seguintes tarefas:

- Mostrar as seguintes informação sobre o jogo: número de jogadas e objetivos a alcançar (por esta ordem).
- Quando o utilizador ganhar ou perder um jogo, devem indicar a vitória ou derrota na janela da aplicação, bem como quando ocorre um baralhamento (*shuffle*).
- Sinalização dos pontos selecionados em cada jogada, utilizando linhas ou quadrados com cores diferentes no fundo.

6 Desenvolvimento do Jogo

Também é fundamental que os alunos cumpram as regras que se seguem no desenvolvimento do jogo.

6.1 Desenvolvimento Faseado

O desenvolvimento deste projeto deverá ser feito de uma forma faseada, devendo os alunos garantir que todas as funcionalidades codificadas até ao momento estão a funcionar corretamente.

É preferível um programa que implementa poucas funcionalidades, mas que funcionam corretamente, do que um programa totalmente desenvolvido mas que não faz nada ou muito pouco.

Assim **sugerem-se** os seguintes passos, pela ordem apresentada, para realização do projeto:

- Inicialização da biblioteca SDL e criação da janela gráfica (código fornecido pelo Prof.).
- Leitura dos parâmetros de funcionamento do programa.
- Geração do tabuleiro inicial.
- Captura dos eventos do rato e processamento para identificação das jogadas.
- Teste de validação de jogadas.
- Atualização do tabuleiro:
    - Remoção dos pontos.
    - Movimentando os restantes pontos.
    - Geração de novos pontos.
- Deteção de quadrados e eliminação de pontos dentro dos quadrados.
- Atualização da interface gráfica.
- Interação com utilizador e reconhecimento das **n** e **q**.
- Cálculo correto da pontuação.
- Escrita na janela do jogo do número de jogadas e os objetivos do jogo.
- Escrita do ficheiro com os resultados dos jogos (estatísticas).
- Implementação de uma das funcionalidades avançadas.

Os alunos deverão garantir a robustez da aplicação verificando todos os casos de erro (por exemplo quando um parâmetro de entrada não seja válido).

6.2 Documentação

O código produzido pelos alunos deverá ser comentado. Os comentários presentes no código deverão explicitar e explicar o funcionamento da aplicação assim como as decisões tomadas. As seguintes regras devem ser cumpridas:

- O código deve ser comentado sempre que realize alguma operação não óbvia.
- Os comentários devem ser claros, gramaticalmente corretos e usando frases simples.
- A declaração de todas as variáveis e constantes deve ser acompanhada de um comentário com uma breve descrição de para que servem.
- Cada bloco de código (seleção ou repetição) deve ser precedido de um breve comentário explicativo.
- Todos os programas devem ter um comentário inicial que identifique, no mínimo, o trabalho, o seu autor, o fim a que se destina e a data de realização.

6.3 Indentação

Um ponto fundamental na organização da escrita de código é a indentação, isto é, organização hierárquica das linhas de código, de acordo com âmbito onde elas se encontram. A indentação deixa o código fonte do programa mais organizado, mais legível, fácil de entender e de modificar, sendo uma parte essencial do trabalho.

6.4 Estrutura do Código

Todos os programas em C devem possuir a mesma estrutura genérica, composta pelas seguintes secções:

- Bloco de comentários.
- Diretivas #include.
- Constantes globais e variáveis globais (caso sejam necessárias).
- Declaração de funções.
- Função main().
- Definição de funções.

Como regra geral, devem considerar que as funções devem caber num único ecrã, isto é, devem ter no máximo cerca de 30 linhas. Também devem cumprir as seguintes regras:

- Inicialize sempre as variáveis na sua declaração.
- Teste a validade dos parâmetros recebidos por uma função.
- Declare constantes e evite usar números no corpo das funções.
- Evite repetições de código, use funções, ciclos, etc.
- Evite o uso de variáveis globais.
- Não use **goto.**
- Escreva código simples e claro que um colega seu possa perceber !

6.5 Compilação

O compilador a usar na execução do projeto é o gcc em ambiente Linux. **Os projetos que não compilem, i.e. que tenham erros de sintaxe, não serão avaliados.** A existência de avisos durante a fase de compilação poderá ser indício da existência de problemas no código. Estes deverão ser eliminados corretamente ou ignorados com cuidado extremo.

6.6 Decisões do Projeto

Como em qualquer projeto de informática, o funcionamento do programa não está totalmente definido no enunciado, existindo algumas ambiguidades e omissões. Para resolver essas omissões os alunos deverão tomar algumas decisões aquando do desenvolvimento do projeto. Estas decisões devem ser fundamentadas, sem nunca ir contra o definido no enunciado.

6.7 Biblioteca SDL

Durante o desenvolvimento deste projeto devera ser usada a biblioteca SDL2. A aplicação deverá ser compilada usando as bibliotecas SDL2, SDL2_image, SDL2_ttf. Mais informação disponível aqui:

- http://wiki.libsdl.org/APIByCategory

- https://wiki.libsdl.org/FrontPage

7 Submissão

Os alunos deverão submeter o código desenvolvido através do sistema FENIX até 15 de Abril. Apenas será necessário entregar o código correspondente ao programa desenvolvido.

Só será aceite um ficheiro de texto com extensão **.c**. Não utilize um processador de texto (e.g. Word) para formatar o seu programa.

8 Plágio

Os trabalhos serão objeto de um sistema de deteção de plágio. Os alunos podem conversar entre si para discutir possíveis soluções para algum problema que tenham mas não podem partilhar código fonte. Nesta entrega intermédia, todo o código deve ser realizado individualmente! Se uma cópia for detetada, os alunos envolvidos na cópia serão penalizados.

9 Avaliação do Projeto

A avaliação do projeto terá em conta diversos parâmetros:

- Funcionalidades implementadas.
- Cumprimento das regras do jogo e das especificações do projeto.
- Estruturação da aplicação, nomeadamente o uso de funções.
- Tratamento de erros.
- Comentários e legibilidade do código.