Jogo de plataformas controlado por acelerómetro

Emanuel Ricardo Brioso, Gabriel Alves de Castro

***Abstract* –** É criado um jogo de plataformas em 2D no qual o salto é controlado pela extensão da perna, permitindo a realização de certos exercícios de reabilitação de forma mais divertida. Esta ideia foi implementada através da utilização de um acelerómetro para ler o movimento da perna, um Arduino nano para confirmar a extensão e a comunicar, e um computador para correr o jogo, no qual o valor enviado pelo Arduino é interpretado como um salto. O acelerómetro e o Arduino estão incorporados num circuito numa breadboard e ligados ao computador por um cabo. De forma a melhor motivar o jogador, contextualizaram-se as ações do protagonista e os níveis que este percorre ao criar a narrativa de que se a personagem é um músico que tem de encontrar o seu microfone para dar um concerto no festival. A programação em Java recorreu à biblioteca jSerialComm e ao LITIENGINE, um game engine desenvolvido por Gurkenlabs.

***Palavras Chave* — Acelerómetro, Arduino, comunicação não bloqueante, comunicação série, game engine, Java, pixelart**

I. Introdução

Quando ocorrem lesões no joelho, o paciente é obrigado a suportar grandes quantidades de tempo sem atividade para além dos exercícios de reabilitação.

Um exemplo ilustrativo é o caso da tendinite patelar, uma condição que provoca dor no joelho e obriga atletas a abandonar competições. A reabilitação é lenta e frustrante, envolvendo exercícios que desenvolvem progressivamente a tolerância a cargas.

Uma das primeiras fases da reabilitação contempla a realização de exercícios isotónicos, o que significa que é aplicada uma carga constante aos músculos e se obriga o paciente a mover as articulações. Isto inclui movimentos de extensão repetida da perna, uma ação repetitiva, o que facilmente se torna aborrecido para pacientes com dificuldade em se concentrarem ou em se motivarem [1].

Este trabalho foca-se na criação de um jogo controlado pela extensão da perna de forma a providenciar entretenimento e motivação para a realização de exercícios de reabilitação. Escolheu-se o género de plataformas porque é intuitivo associar o movimento de extensão à ação de salto, o que providencia uma melhor imersão no jogo.

II. Métodos e Materiais

A. *Circuito*

Para este trabalho foi criado um circuito numa *breadboard* no qual se liga o acelerómetro MMAB452 ao Arduino nano por quatro fios (figura 1).

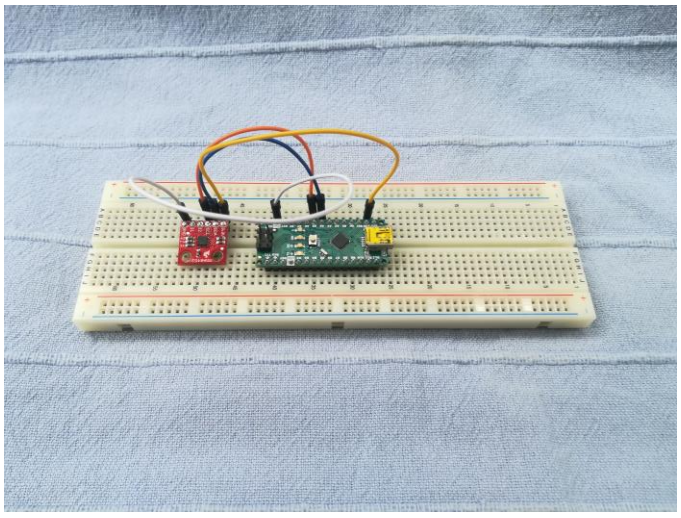


Fig. 1 – Circuito que liga o acelerómetro ao Arduino.

Trata-se de um circuito fácil de implementar que utiliza o pino do Arduino nano com output de 3,3V de forma a não sobrecarregar o acelerómetro. O design recomendado contemplaria a utilização de resistências para permitir maior segurança, mas devido à impossibilidade de as adquirir estas não foram implementadas [2].

Finalmente, o Arduino liga-se ao computador por um cabo USB.

B. *Interface Arduino - Java*

Para que o jogo funcione corretamente, é necessário o funcionamento em simultâneo do acelerómetro, do Arduino e do computador de acordo com o diagrama da figura 2.

Definiu-se que o movimento horizontal do protagonista seria controlado pelas setas do teclado.

O salto começa com a extensão da perna do jogador, o acelerómetro lê a aceleração desse movimento e transmite esses dados ao Arduino, onde o programa calcula se a aceleração foi suficiente para justificar que ocorra o salto.

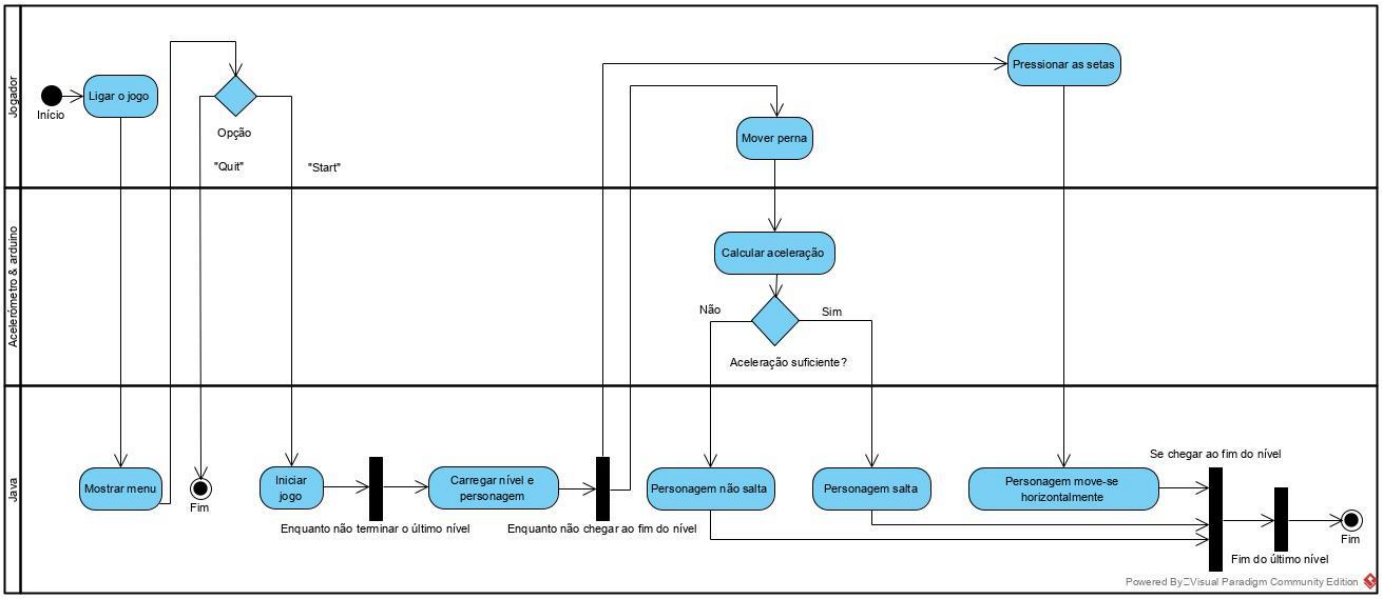


Fig. 2 – Diagrama de atividades para o funcionamento do jogo.

Só quando o movimento é realizado corretamente é que o Arduino comunica ao computador a informação de que o salto deve ser realizado, o que se traduz no movimento vertical da personagem que o jogador controla.

Para a definição da extensão de perna no Arduino foi usado um intervalo de acelerações adequado ao movimento que era pretendido tal como se pode observar na figura 3.

```
void loop() {
  if (accel.available()) {      // Wait for new data from accelerometer
    previousAcStatus = acStatus;

    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis >= interval) {
      previousMillis = currentMillis;

      acStatus = abs(accel.getCalculatedZ());

      if (previousAcStatus < 0.5 && acStatus > 0.6) {
        Serial.print('1');
      }
      else {
        Serial.print(' ');
      }
    }
    else {
    }
  }
```

Fig. 3 – Código da secção *loop()* do Arduino.

Por fim nesta interface entre o Arduino e o Java, foi necessário evitar o uso da função *delay* para que a comunicação fosse não bloqueante, caso contrário o jogo não iria correr enquanto não fossem lidos os valores do acelerómetro.

Logo, para implementar o código de Arduino foi feito um incremento de tempo no *loop()* que apenas captava os dados do acelerómetro num invervalo de 300 *ms*. Este intervalo serve para se registar uma mudança significativa de aceleração quando a perna é extendida.

Após ser registado uma extensão de perna correta, é enviado um *char* '1' através da comunicação série do Arduino.

Para a captação deste *char* na componente Java, é necessário recorrer à biblioteca *JSerialComm* que permite também fazer uma comunicação não bloqueante [3].

Através das funções desta biblioteca foram lidos os bytes que são transmitidos através da porta série e transformados em *char*. Caso o *char* corresponda a '1', então será invocada a função *jump()* que se traduz no salto do personagem do jogo.

Esta comunicação é atualizada constantemente através do game engine como é mencionado na secção seguinte.

```
@Override
public void update() {
    // reset the number of consecutive jumps when touching the ground
    if (this.isTouchingGround()) {
        this.consecutiveJumps = 0;
    }

    triggerHandler();

    try {
        if (comPort.bytesAvailable() == 0) {

        } else {
            byte[] readBuffer = new byte[comPort.bytesAvailable()];
            int numRead = comPort.readBytes(readBuffer, readBuffer.length);
            currentAcStatus = (char) readBuffer[numRead - 1];
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    if (currentAcStatus == '1') {
        this.jump();
        currentAcStatus = '0';
    }
}
```

Fig. 4 – Comunicação Arduino – Java no código de Java

C. *Programação em Java*

Na figura 5, pode-se observar o diagrama de classes geral do projeto. As classes a verde pertencem ao *game engine* LITIengine [4] e as classes azuis são as classes criadas especificamente para este jogo.

Existem duas classes que herdam da classe *Screen*, que são a classe *MenuScreen* e a *IngameScreen*. A classe que gere o menu principal, *MenuScreen*, começa por inicializar as imagens e música do background. A classe *IngameScreen* carrega o nível selecionado e chama a função *instance()* da classe *Player*, esta função verifica se já existe um objeto *Player* a ser processado no ecrã, caso contrário este é criado e colocado nas coordenadas especificadas como sendo de *spawn* do respetivo nível.

A classe *Player* trata de criar o jogador e captar input tanto do teclado como do acelerómetro, caso este esteja ligado. Cada objeto *Player* tem um atributo do tipo *Jump* que serve para acionar o salto do personagem.

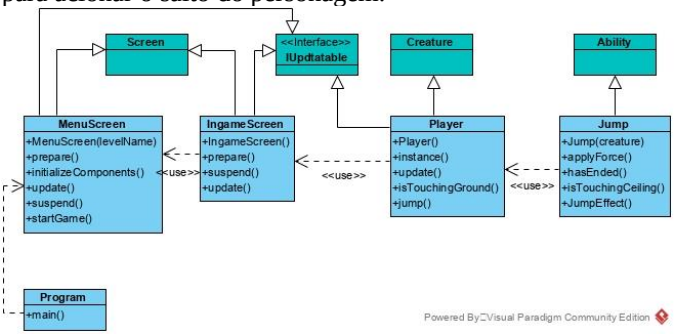


Fig. 5 – Diagrama de Classes do código em Java.

É importante mencionar que as classes *MenuScreen*, *IngameScreen* e *Player* são atualizadas a cada momento pelo game engine, pois implementam a interface *IUpdateable* e a cada atualização executam a sua função *update()*.

D. *Design*

O contexto que se deu à ação que se desenrola no jogo foi de que o protagonista é um músico que outrora fora famoso e agora quer encontrar o seu microfone para dar um concerto num festival de verão. Para tal terá de ultrapassar os obstáculos da floresta vizinha ao festival, das tendas do público e das traseiras dos palcos.

Fig. 6 – Sprites do protagonista parado, a andar e a saltar.

Para criar a pixelart das animações do protagonista e dos níveis foi utilizada a ferramenta online Pixilart (www.pixilart.com). A imagem do menu inicial foi criada no Microsoft Paint.

III. Resultados e Discussão

Ao correr o jogo, é apresentado em primeiro lugar, um menu que dá para selecionar o nível que o jogador irá jogar, assim como um botão *"Exit"* que faz com que o programa feche.

Fig. 7 – Menu inicial do jogo.

Após selecionar um dos três níveis, as texturas do respetivo nível são processadas e a entidade do jogador é inicializada e colocada no ponto de "*spawn*".

Fig. 8 – Jogador a saltar no primeiro nível.

O objetivo do jogador é atravessar as várias plataformas dos níveis usando o acelerómetro firme na perna para poder realizar uma extensão de perna.

Fig. 9 – Exemplificação de extensão de perna.

De todos os testes que foram feitos ao longo do projeto e depois de ajustar os limites de aceleração que acabaram por definir uma extensão de perna, observou-se que na posição correta, todas as extensões de perna são detetadas pelo código de *Arduino*.

Algumas sugestões para trabalhos futuros e acréscimos que ajudariam a melhorar o objetivo deste projeto são discutidos na conclusão.

IV. Conclusão

Tendo em conta o resultado, que é um jogo em que uma extensão de perna correta do jogador resulta num salto virtual num jogo de plataformas, observou-se um alto potencial para que seja usado como uma ferramenta em tratamentos de certas patologias, por exemplo, tendinite patelar.

Em termos de melhoramento e futuras funcionalidades, seria interessante adicionar comunicação extra com o jogador, por exemplo, texto que motivasse a sua recuperação ao longo da sessão.

E, para se abordar um maior número de exercícios podia-se generalizar ou disponibilizar diferentes configurações pré-feitas para diferentes tipos de exercícios de reabilitação.

Por fim, para aumentar a usabilidade e conforto do paciente, dever-se-ia ajustar o hardware de modo a que o acelerómetro fosse confortável de usar, transformando o sistema num *wearable*.

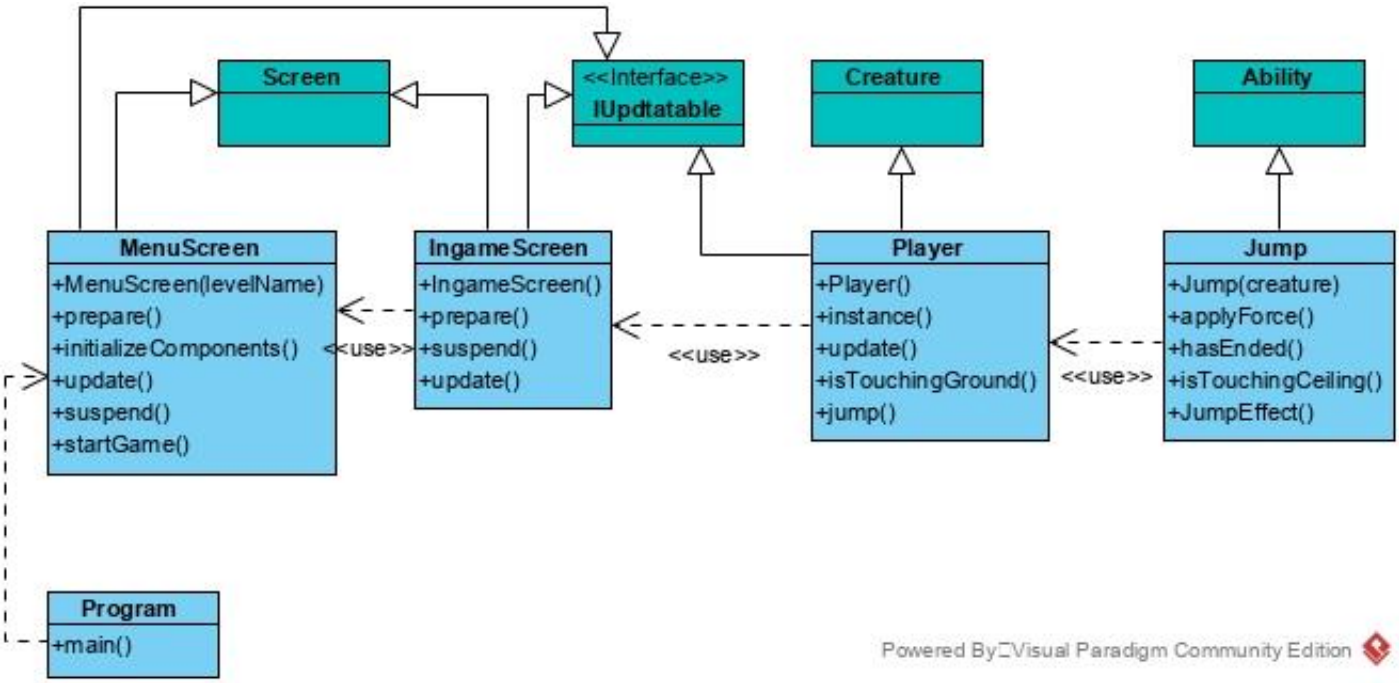
Agradecimentos

Queremos agradecer aos professores docentes de LIEB, que foram acompanhando o nosso trabalho e sugerindo conselhos úteis. Agradecemos também aos Gurkenlabs, por disponibilizarem de graça o seu game engine apropriado a jogos 2d em Java.

Por fim, agradecemos ao laboratório e o seu técnico por disponibilizarem o material que usamos para tornar o projeto possível.

Referências

1. Malliaras, P., et al., *Patellar Tendinopathy: Clinical Diagnosis, Load Management, and Advice for Challenging Case Presentations.* The Journal of orthopaedic and sports physical therapy, 2015. **45**: p. 1-33.
2. *MMA8452Q Accelerometer Breakout Hookup Guide*. Available from: https://learn.sparkfun.com/tutorials/mma8452q-accelerometer-breakout-hookup-guide/all#example-hookup.
3. JSerialComm: https://fazecast.github.io/jSerialComm/
4. Github LITIengine: https://github.com/gurkenlabs/litiengine



Anexo 1 – Diagrama de classes em Java