Stag: The Savior

http://labmm.clients.ua.pt/LM3/LM3-50/

João Sanches, Luís Cardoso, Maria Oliveira, Simão Bentes
80490, 99601, 99356, 97761
{jlmpsanches@ua.pt, gmadail@ua.pt, mendes.oliveira@ua.pt, bentes@ua.pt}

1. Introdução

Para a elaboração do projeto final da unidade curricular de Laboratório e Multimédia 3, foi proposto ao grupo pela *Gamers4Nature* desenvolver um jogo em Javascript onde fosse retratado o tema da espécie d "vaca-loura". A vaca-loura ou *stag beetle* é um inseto que atualmente se encontra em vias de extinção devido à destruição descontrolada do ecossistema por parte de indústrias madeireiras. Assim, o grupo optou por aplicar o tema de uma forma mais cómica, através da criação de uma rivalidade entre estes insetos e lenhadores com o objetivo de chegar ao maior número de pessoas. Embora seja aplicada uma conotação cómica, pretendemos transmitir uma mensagem de alerta com o intuito de fazer o jogador refletir sobre este tema e repensar os seus hábitos.

1.1. Conceito e categorização do produto multimédia

Definimos como personagem principal do nosso jogo o *Stag*. *Stag* é um escaravelho-veado que tem como missão proteger a sua colónia, que se refugia na árvore-mãe, de Lenhadores que tentam chegar à floresta a todo o custo. O nosso personagem persegue e esforça-se para impedir que a indústria madeireira consiga atingir o seu objetivo e destruir o habitat dos escaravelho-veado.

Para o protagonista, o tempo é um fator crucial na sua missão. À medida que o jogo decorre, há uma barra que vai diminuindo e só aumenta quando os familiares do *Stag* se conseguem abrigar. Quando esta barra fica vazia, o jogo é dado como terminado. A dificuldade do jogo aumenta com o desenrolar da partida.

O nosso produto é do Plataforma 2D, apenas com a possibilidade de jogar em modo *single player*, e a sua estética parte do movimento artístico de *Pixel Art*. Em relação à sua categoria, é de ação e estratégia, tendo em conta o seu conceito de conseguir ultrapassar desafios até alcançar o objetivo final.

1.2. Justificação da escolha do tema

Atendendo à sugestão feita pela Games4Nature, decidimos abordar o tema sobre um inseto denominado de vaca-loura que se encontra em vias de extinção por várias razões, nomeadamente, a intensidade de incêndios e a urbanização e monocultura industrial que levam à destruição do seu habitat. Desse modo, optamos por nos focar numa dessas mesmas causas e abordá-la de forma mais aprofundada para que conseguíssemos transmitir uma mensagem a cada jogador. As indústrias madeireiras foram as selecionadas para o inimigo da nossa personagem principal (um vaca-loura) para que essa mesma mensagem que se pretendia passar se tornasse mais clara e percetível e, por outro lado, captasse a atenção e a sensibilidade de cada jogador.

1.3. Objetivos gerais

Acreditamos que, com a elaboração deste projeto final, os nossos conhecimentos de *Javascript* evoluam. Foi assim que aconteceu com as duas cadeiras de Laboratório anteriores a esta. Os respetivos projetos finais capacitaram-nos de ferramentas que só as aulas por si só não nos forneceram. Apesar de sabermos a importância das aulas práticas, que têm um papel fundamental na introdução a uma nova linguagem, somos defensores de que o autoconhecimento pode ser um método ainda mais eficaz. Através da pesquisa nas documentações da linguagem e, principalmente, através da tentativa erro.

Pretendemos também que o projeto final seja visto como um produto: um verdadeiro jogo. Não queremos de forma alguma que se sinta que o projeto seja a representação da matéria lecionada. Queríamos inovar, criar um jogo limado e com atenção aos pormenores e que se sentisse profissionalismo. Esta é uma das razões pela qual escolhemos o *Canvas*, por sabermos ser o *standart* da implementação de jogo web.

Funções de jogo mais específicas, como guardar informações de jogo após fechar o browser, o jogo ser responsive e adaptar-se ao tamanho da janela e mesmo a possibilidade de *fullscreen* sempre estiveram como objetivos de implementação. Apesar de serem pormenores que não a afetam a jogabilidade do jogo, acreditamos que são estas pequenas características que fazem a diferença.

2. Planeamento e Pré-produção

2.1. Brainstorming

Partindo do tema que nos foi proposto pela Gamers4Nature, o inseto Vaca-Loura, inicialmente optamos por um produto que tivesse como objetivo algo mais cómico de forma que pudesse atingir um público-alvo mais abrangente para que, dessa forma, conseguíssemos alcançar o maior número de pessoas possível. Por outro lado, um dos nossos propósitos seria que o jogo transmitisse uma mensagem de esperança e, ao mesmo tempo, que fizesse com que os nossos jogadores

repensassem nos seus hábitos enquanto elementos de uma sociedade.

O inseto Vaca-Loura, denominado de Stag, tinha como missão proteger a sua colónia de Lenhadores inimigos que tentariam chegar à floresta incessantemente. Para evitar que isso pudesse acontecer, a personagem principal perseguiria e tentaria impedir que a indústria madeireira entrasse e destruísse o seu habitat.

Nesta tarefa, o fator tempo era algo crucial no sentido em que se o jogador não conseguisse concluir o seu objetivo com sucesso dentro do intervalo de tempo estabelecido, este perderia. A dificuldade desta aventura aumentaria progressivamente à medida que o jogador conseguisse concluir cada nível com sucesso.

Em relação à jogabilidade este seria apenas do modo *single player* e do tipo Plataforma 2D. A sua estética partiria do movimento artístico de *Pixel Art* e a sua categoria seria de ação e estratégia, tendo em conta o seu conceito inicial de conseguir ultrapassar vários desafios até alcançar o objetivo final.

2.2. Público-alvo

O nosso público-alvo acarreta um intervalo de valores dos oito aos vinte. A escolha deste público-alvo deve-se à vontade, de nossa parte, de conseguirmos alcançar o maior número de pessoas possível para que estas se pudessem divertir com o jogo e, por outro lado, se sentissem sensibilizadas para repensarem nos seus maus hábitos enquanto elementos da sociedade.

2.3. Investigação e seleção de conteúdos

Para a personagem principal deste projeto, foi escolhido o inseto vaca-loura e foi denominado de Stag. Como personagens secundárias, foram selecionados os familiares do Stag que irão tentar refugiar-se na "árvore-mãe". Por último e como inimigos do jogador, os Lenhadores têm como objetivo desflorestar o território onde habitam os vaca-loura não se importando de passar por cima dos mesmos.

Em relação ao cenário, é constituído por um terreno, por um céu, pela floresta e pela "árvore-mãe". É composto também por nuvens que se irão movimentar ao longo do tempo. Todos estes objetos foram criados em diferentes ficheiros de forma a criar camadas de exibição entre eles, no *Canvas*, sem que exista uma "quebra" na colisão. Inicialmente tínhamos definido que quando as baratas colidissem com a árvore voltavam à sua posição aleatória fora da área de jogo. No entanto sentimos que ao colidir com a árvore e a desaparecer instantaneamente, não ficava esteticamente muito bem. Criámos assim uma cópia da área de jogo onde as baratas entram para podermos passa-las por baixo da árvore sem que se vejam ou que exista a tal "quebra". Referimo-nos ao ficheiro "zona_entraga.png", na pasta *imgs*.

No decorrer do jogo, irão aparecer algumas moedas espalhadas pelo terreno. Estas servirão para que, futuramente, o jogador consiga adquirir evoluções de velocidade para a personagem principal e de tempos de Power Up's.

2.4. Arborescência

A arborescência do nosso jogo está em anexo, na Figura 1. Quando o jogo é aberto, deparamo-nos com a Página Inicial do mesmo onde conseguimos ter acesso a três botões diferentes: Jogar, Instruções e Sobre. Por meio do botão Jogar iniciamos o próprio Jogo. Por outro lado, através do botão Instruções ingressamos numa janela onde podemos ver algumas Instruções em relação à forma como se joga, nomeadamente, às teclas utilizadas durante o jogo. Por último, por intermédio do botão Sobre temos a possibilidade de perceber quem são os vários elementos contribuintes para a elaboração deste projeto, bem como, uma pequena sinopse acerca o mesmo. Na área de jogo é possível aceder a um botão que tem como função a passagem para o modo ecrã inteiro. Nesse mesmo local, quando o jogador perde a partida, passará automaticamente para o ecrã intitulado de Game Over. Por fim, existe também um botão que nos redireciona para o Menu de Pausa. Nesse mesmo Menu, conseguimos tirar/meter o som dos efeitos sonoros e, também, a música de fundo do jogo. Também é possível aceder à Loja onde é possível comprar alguns power up's com as moedas que vão sendo coletadas durante o decorrer do jogo. Quando já não se quer jogar mais, pode sair-se do jogo a qualquer momento através do mesmo Menu de Pausa.

3. Produção

3.1. Estudo da interface

Na figura 2, em anexo, conseguimos ver o logo e ter a perceção dos conteúdos que estão presentes na página inicial do jogo. Através desta mesma página é possível aceder ao jogo, às instruções do mesmo e, por último a uma breve narração sobre o jogo.

Através da figura 3, vemos a área e o cenário do jogo, tal como as várias personagens presentes. Observa-se também uma barra correspondente ao tempo restante de jogo que vai diminuindo ou aumentando consoante a prestação do jogador. Podemos verificar que também na imagem 3, é-nos apresentada a quantidade de moedas que temos disponíveis, o acesso ao menu Pausa através do botão que se encontra no canto superior direito, a contagem do tempo decorrido e a opção de fullscreen no botão que está posicionado no canto inferior direito.

A figura 4 contém o meu de pausa, onde podemos distinguir as várias opções. Temos a possibilidade de ligar e desligar o som da música de fundo bem como o som e os efeitos sonoros presentes no jogo. Dá-nos também a possibilidade de ter acesso à loja onde podemos comprar Power Up's.

Na figura 5, podemos concluir que nos são apresentados todos os controlos que o jogo tem, assim como as teclas para cada controlo e cada funcionalidade. Para além disso, apresenta-nos também a descrição de

todas as personagens intervenientes no jogo, e dos elementos presentes na área do mesmo.

3.2. Fluxogramas

Em anexo temos dois fluxogramas que representam mecânicas importantes no nosso jogo. O primeiro é um exemplo de uma colisão. O segundo representa a colocação inicial dos lenhadores de forma aleatória. (Ver Figuras 8 e 9)

3.3. Estética e softwares utilizados

A estética do nosso jogo é baseada num estilo pixelizado e, para que tal fosse possível, procurámos a utilização de softwares adequados e que nos ajudassem na sua realização.

A área de jogo (nomeadamente, a relva e o céu) foi criada através do Adobe Photoshop. Em relação às personagens e a alguns outros elementos, foram utilizados o Aseprite[1] (para a elaboração dos Lenhadores) em conjunto com o website pixilart.com[2] (para a produção dos familiares e o suco de seiva). Posteriormente, muitos destes componentes, principalmente as personagens, foram reutilizados no vídeo de promoção que foi conceptualizado no Adobe Premiere Pro (figura 10), sendo que em alguns houve a necessidade de edição e adaptação antecipada.

É de salientar que vários elementos do jogo passaram por múltiplas etapas até ser obtido o resultado final (figura 11). Por outro lado, foram também descartadas algumas possíveis ilustrações que não se enquadravam no ambiente (figuras 12 e 13) ou pela sua movimentação não se assemelhar às demais. (figura 14).

3.4. Formatos dos media

Todos os media que foram utilizados possuem formatos adequados que visam a facilitação na programação e implementação do jogo. As imagens (da área de jogo assim como as personagens) encontram-se em formato PNG. Em relação os áudios estão todos no formato MP3 e o vídeo promocional em MP4.

3.5. Implementação técnica

Para todas as colisões do jogo, utilizamos uma fórmula para a colisão entre dois retângulos, que encontramos na documentação da Mozilla[3]. No entanto, o facto de existirem diferentes tipos de colisões com diferentes tipos de abordagens, não nos facilitou assim tanto a tarefa. Um exemplo disso é a colisão do Stag com qualquer elemento. O facto de, na horizontal, mudar de *sprite* faz com que as colisões também mudem. Tivemos de "recortar" a área de colisão dependendo da posição do jogador no momento. (Ver Figura 15)

Terminado este processo, deparámo-nos com um problema ao nível da jogabilidade: se a nossa personagem não se movesse, as colisões iriam se manter e o jogador iria tirar partido disso, algo que não queríamos. Tivemos de fazer com que os lenhadores se desviem, se o Stag estiver parado. Adicionamos então uma nova colisão ao jogo, onde se o lenhador colidisse com o lado esquerdo do stag, ou seja, o x do stag com a sua largura mais uma margem de distância de 200, parava o movimento horizontal e movia-se verticalmente até a colisão deixar de existir. Tivemos ainda de definir se o lenhador se deslocaria para cima ou para baixo, quando se desvia da personagem. Criámos uma condição onde se a metade do corpo do lenhador for menor que a metade do jogador, o lenhador sobe, se for maior, desce.

Inicialmente o nosso jogo tinha as nuvens estáticas, inseridas no background. No entanto, após criar o movimento do lenhador, sentimos que um potencial movimento das nuvens não seria assim tão diferente do código já utilizado. Quando o objeto sai da área de jogo é novamente posicionado de forma aleatória no lado oposto. A maior diferença estava no tamanho e velocidade que as nuvens pudessem vir a ter. Tínhamos a noção de que não ficaria visualmente muito apelativo as nuvens serem apresentadas todas com o mesmo tamanho. Assim, criámos a propriedade tamanho no objeto nuvem. Este tamanho vai ser gerado de forma aleatória sempre que a nuvem é reposta. Multiplicado pelo tamanho base da imagem é possível obtermos tamanhos diferentes de nuvens. Para terminar, nuvens mais pequenas vão-se deslocar mais rapidamente. O método mover() do objeto nuvem pode ser visto em detalhe a figura 16.

Durante a planificação deste projeto, propusemo-nos a utilizar o requestAnimationFrame() para animar o nosso jogo. Ao aplicarmos este método, percebemos que fazia com que o jogo se comportasse de maneira diferente de utilizador para utilizador, e não é isso que queremos. A razão pela qual isto acontece prende-se com o facto de existirem diferentes taxas de atualização nos ecrãs. Em dois computadores, um com capacidade de reproduzir 144fps e outro 30fps, o jogo decorrerá mais rápido num que noutro. Assim, tivemos de definir uma taxa de atualização, igual para todos. Baseamo-nos no código do GitUser Elund Mark e adaptámos à nossa situação. Podíamos ter ido por um caminho mais fácil e criar um timer que repetisse 1000/60ms, por exemplo, mas escolhemos seguir a opção mais apropriada (na opinião de Elund Mark e outros programadores). O nosso jogo vai ser assim apresentado a 60 frames por segundo. No entanto, os sprites das nossas personagens vão ser animados a 10fps, pois apesar de querermos que as personagens se movimentem de forma fluida, as 4 ou 6 imagens de um sprite iriam ser apresentadas demasiado rápido.

A utilização do Canvas para a implementação do nosso jogo tanto nos trouxe vantagens, como algumas desvantagens. Um exemplo disso foi a dificuldade que tivemos em implementar um simples click. O facto do <canvas> ser só um elemento e todos os botões dentro do jogo serem apenas "desenhos", leva-nos a podermos apenas utilizar o onclick em relação a todo o <canvas>. A única abordagem encontrada foi a deteção das coordenadas do click e, desta forma, se essas coordenadas se encontrarem dentro de uma determinada área, o utilizador clicou no botão, por exemplo. O event.pageX e

o event.pageY[4] retornam-nos os valores horizontais e verticais no momento do click. Para complicar todo este processo, como o jogo se adapta à janela do ecrã e varia de tamanho, também os botões do jogo alteram as suas coordenadas em relação à janela do utilizador. Tivemos então de nos afastar dos valores fixos e passarmos a percentagens. Em vez de identificarmos se o click se encontra entre, por exemplo, os 100px e os 200px, temos de identificar se ele se encontra entre os 10% e o 15% da área do <canvas>.

Para além de termos o nosso jogo responsive, tínhamos como objetivo que fosse possível jogá-lo em fullscreen, tornando-o mais imersivo. A documentação W3C deu-nos a conhecer o método x.requestFullscreen()[5], onde x é o elemento que queremos colocar em ecrã inteiro. O botão criado com este propósito iria alterar o ícone dependo do estado do jogo no ecrã ("aumentar tela", quando não estivesse em fullscreen e "diminuir tela" quando estivesse). Inicialmente caímos no erro de este botão ser alterado quando fosse clicado e esquecemo-nos que poderia existir a possibilidade de o jogador sair do ecrã inteiro através da tecla "esq" (algo pré-definido pelo browser) e isso desconfigurar a ordem em que o ícone é apresentado no botão. Assim, a propriedade document.fullscreenElement[6] permite saber se a nossa página se encontra em fullscreen e assim configurar o sprite do botão em relação a isso.

A pontuação que o jogador obtém é o que define a sua performance. De forma a manter o interesse do jogador, quisemos que este se desafiasse ao superar a sua pontuação máxima. Assim, procurámos uma forma em Javascript de guardar a pontuação máxima do utilizador, mesmo que este feche o browser e saia do jogo. Foi no Stack Overflow que encontrámos a resposta: o localStorage[7]. Desta forma, podemos guardar informações e voltar a utilizá-las posteriormente, mesmo que o browser seja fechado. Estes dados ficam sempre guardados. O mesmo método também foi utilizado para armazenarmos as moedas do utilizador. Esta abordagem, na nossa opinião, contribui para que o jogo seja mais desafiante e que possa ser jogado várias vezes sem que se perca o progresso. Se assim não fosse, sempre que o jogador quisesse adquirir algo na loja, teria que conseguir arrecadar todas as moedas necessárias em cada jogo ao invés de as armazenar partida após partida.

No final do jogo, queríamos também que existisse um botão para o jogador poder partilhar a sua pontuação na rede social Twitter. Após uma pesquisa pela documentação oficial da empresa, na secção dos botões Twitter[8], percebemos que existia um padrão de link que permitia criar um tweet pré-definido, pronto a publicar. No entanto, este padrão não aceita todo o tipo de caracteres, como o "#" para criarmos hashtags relativas ao jogo, por exemplo. O problema resolveu-se ao encontrarmos um enconder de texto para URL[9], onde facilmente "traduzimos" o que queríamos escrever em forma de link. Para terminar, tivemos de concatenar a pontuação com o link criado, de forma a que cada utilizador tivesse um tweet personalizado com a sua pontuação. Como o <canvas> é um elemento que não contém qualquer tipo de outros elementos, não é possível adicionar o elemento <a> para nos redirecionar para outras páginas. Foi na documentação da W3C que encontrámos uma solução bastante simples: o método window.open()[10] que, no fundo, tem o mesmo papel que um href, onde nos abre um novo separador com página que definimos no parâmetro. (Ver Fig. 17)

4. Resultados

4.1. Análise crítica

Em discussão com todos os elementos do grupo chegamos à conclusão de que, apesar de terem surgido alguns problemas, estamos bastante satisfeitos com o produto final. Conseguimos que o jogo atenda a um caráter educativo esperado e, desse modo, alcançar o sucesso do desafio que nos foi proposto pela Gamers4Nature.

Apesar de alguns contratempos que surgiram no processo de desenvolvimento, pensamos que o produto final é do orgulho de todos os membros do grupo. Sentimos que a ideia inicial que tínhamos para este jogo foi alcançada, sem que a programação nos limitasse e, muitas vezes, fosse ela a impulsionadora de novas ideias e decisões para o jogo. O balanço é, assim, positivo.

5. Trabalho Futuro

Com vista a futuras atualizações, pensámos na possibilidade de implementação de algo que possa vir a tornar o jogo mais personalizável. Como exemplo disso, o jogador poderá personalizar a capa do Stag ou, até, modificar o cenário do próprio jogo (como adicionar neve ou colocar em modo noturno). Por outro lado, gostaríamos também que o jogo estivesse adaptado a todo o tipo de dispositivos. Tudo isto, a fim de tornar o jogo mais dinâmico e apelativo.

6. Referências Bibliográficas

[1] Aseprite: pixel art design software
https://www.aseprite.org/
[2] Pixilart: pixel art design website
https://www.pixilart.com/
[3] 2D Collision Detection
https://developer.mozilla.org/en-US/docs/Games/Techniques/2D_collision_detection
[4] w3schools.com: MouseEvent pageX Property
https://www.w3schools.com/jsref/event_pagex.asp
[5] w3schools.com: How TO – Fullscreen
https://www.w3schools.com/howto/howto_js_fullscreen.asp

[6] w3schools.com: HTML DOM fullscreenElement Property
https://www.w3schools.com/jsref/prop_document_fullscreenelement.asp

[7] stack overflow: Saving a variable after closing browser
https://stackoverflow.com/questions/29310332/saving-a-variable-after-closing-browser

[8] Tweet button
https://developer.twitter.com/en/docs/twitter-for-websites/tweet-button/overview
[9] URL Decoder/Encoder
https://meyerweb.com/eric/tools/dencoder/
[10]w3schools.com: window.open()
https://www.w3schools.com/jsref/met_win_open.asp
[10] Efficiently load JavaScript with defer and async: async and defer
https://flaviocopes.com/javascript-async-defer/#async-and-defer
[11] GitBun Gist: Controlling the Frame Rate with requestAnimationFrame
https://gist.github.com/elundmark/38d3596a883521cb24f5

7. Anexos

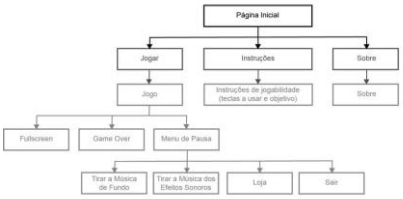

Fig. 1 – Arborescência

Fig. 2 – Página Inicial

Fig. 3 – Área de Jogo

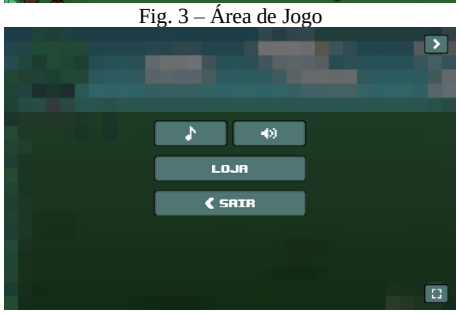

Fig. 4 – Pausa

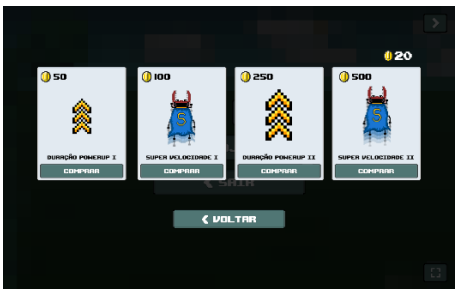

Fig. 5 – Loja

Fig. 6 – Instruções

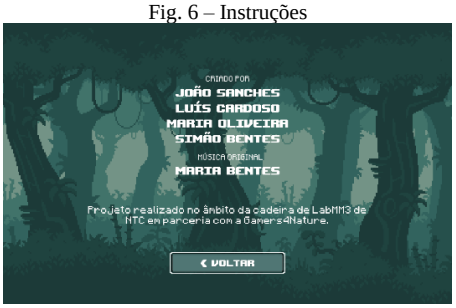

Fig. 7 – Sobre

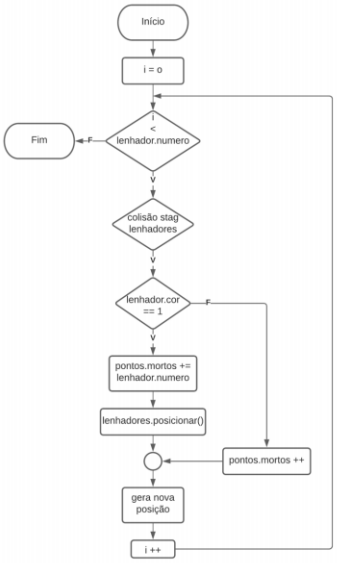

Fig. 8 – Colisão Stag com os Lenhadores

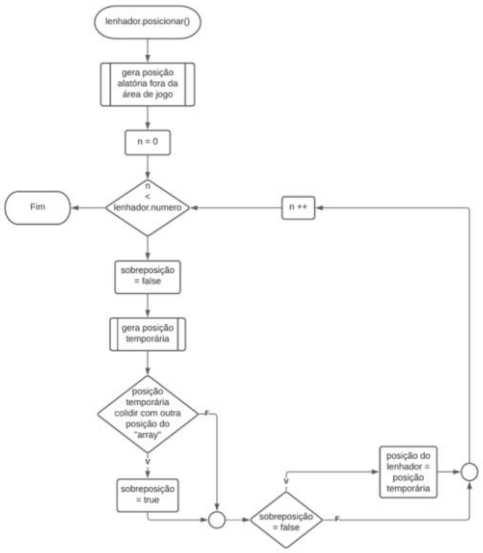

Fig. 9 – Posicionar Lenhadores sem existir sobreposição

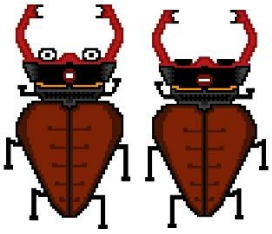

Fig. 10 – Posicionar Lenhadores sem existir sobreposição

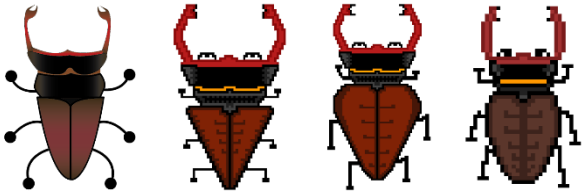

Fig. 11– Evolução da conceptualização dos familiares

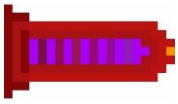

Fig. 12 – Tiro do Power Up não utilizado

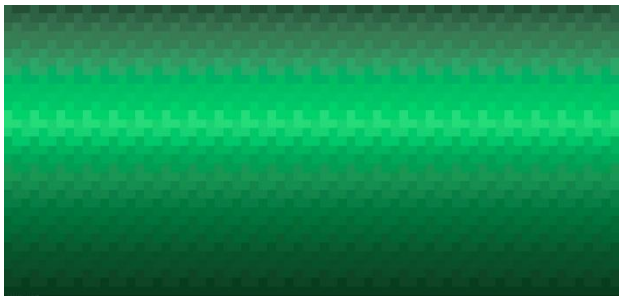

Fig. 13 – Proposta de relva para o jogo

Fig. 14 – Opções descartadas do Stag na lateral

```
//Queremos que a colisão do stag mude dependendo do sprite ativo
if (jogador.spriteY === 0 || jogador.spriteY === 3) {
    sprite_x_removido = 25
    sprite_y_removido = 0
} else {
    sprite_x_removido = 0
    sprite_y_removido = 25
}

if (jogador.x + sprite_x_removido < lenhador.x[i] + lenhador.largura &&
    jogador.x + jogador.largura - sprite_x_removido > lenhador.x[i] &&
    jogador.y + sprite_y_removido < lenhador.y[i] + lenhador.altura &&
    jogador.y + jogador.altura - sprite_y_removido > lenhador.y[i]) {
```

Fig. 15 – Alterar colisões conforme a orientação do *Stag*

```
mover: function () {
    //nº de nuvens definido pelo nuvens.num
    for (let i = 0; i <= this.num; i++) {
        //Desenho das imagens
        c.drawImage(this.img, this.x[i], this.y[i], this.lAleatoria[i], this.aAleatoria[i]);
        //enquato é visivel na área de jogo, a nuvem anda
        if (this.x[i] > -150) {
            this.x[i] -= this.vel[i]
        } else {
            //gera-se uma nova posição
            this.x[i] = nAleatorio(jogo.width, jogo.width + 1000)
            this.y[i] = nAleatorio(0, 130)
            //define-se um novo tamanho. Este valor vai ser multiplicado pelo tamanho base da nuvem
            this.tamanho = Math.random() * 0.45 + 0.35
            this.lAleatoria[i] = this.largura * this.tamanho
            this.aAleatoria[i] = this.altura * this.tamanho
            //é gerada uma velocidade aleatório tendo em conta o tamanho
            //nuvens mais pequenas andam mais depressa
            if (this.tamanho < 0.35) {
                this.vel[i] = Math.random() * 0.2 + 0.15
            } else if (this.tamanho < 0.4) {
                this.vel[i] = Math.random() * 0.15 + 0.1
            } else {
                this.vel[i] = Math.random() * 0.1 + 0.05
            }
        }
    }
    //se o powerup 1 estiver ativo, todas a nuvens vão ter a vel 40
    if (powerup.ativo && powerup.tipo === 1)
        nuvem.vel.fill( value: 40)
}
```

Fig. 16 – Método mover para a nuvem

Fig. 17 – Tweet pré-definido