UNIVERSIDADE FEDERAL DE SÃO CARLOS

LABORATÓRIO DE ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES 2

PROF. DR. LUCIANO DE OLIVEIRA NERIS

FELIPE CHURUYUKI CHINEN – 496235
Engenharia de Computação
IGOR GUILHERME BIANCHI – 558400
Engenharia de Computação

RELATÓRIO 3

Descrição completa do jogo "DodGG"

SÃO CARLOS
2017

SUMÁRIO

LISTA DE ILUSTRAÇÕES E TABELAS

1 INTRODUÇÃO

Este relatório tem como objetivo apresentar o projeto do jogo “DodGG” desenvolvido pelos alunos Igor G. Bianchi e Felipe C. Chinen durante a disciplina de Laboratório de Arquitetura e Organização de Computadores 2.

A próxima seção deste capítulo contará a breve história dos jogos eletrônicos, começando por projetos realizados em universidades ao redor do mundo, passando por consoles e jogos clássicos e chegando até os dias atuais, com placas gráficas poderosas e influência cultural gigantesca, movimentando milhões de dólares anualmente. O destaque fica por conta de que os games invadiram as vidas das pessoas e são parte importante da economia e cultura atualmente.

O segundo capítulo, em sua primeira seção, ficará responsável por explicar completamente o desenvolvimento do jogo, suas principais rotinas e variáveis. A segunda seção irá tratar dos elementos gráficos e sua implementação. A terceira e última seção deste capítulo, se encarrega de explicar a divisão de tarefas entre os membros do grupo de desenvolvimento.

O terceiro capítulo irá relatar as funcionalidades do jogo, além de suas respectivas telas. Também será apresentado o diagrama de estados do game, bem como sua explicação.

O quarto e último capítulo será o responsável por discutir os problemas apresentados durante o desenvolvimento do projeto, dizer o que era esperado no começo e qual foi o resultado no fim.

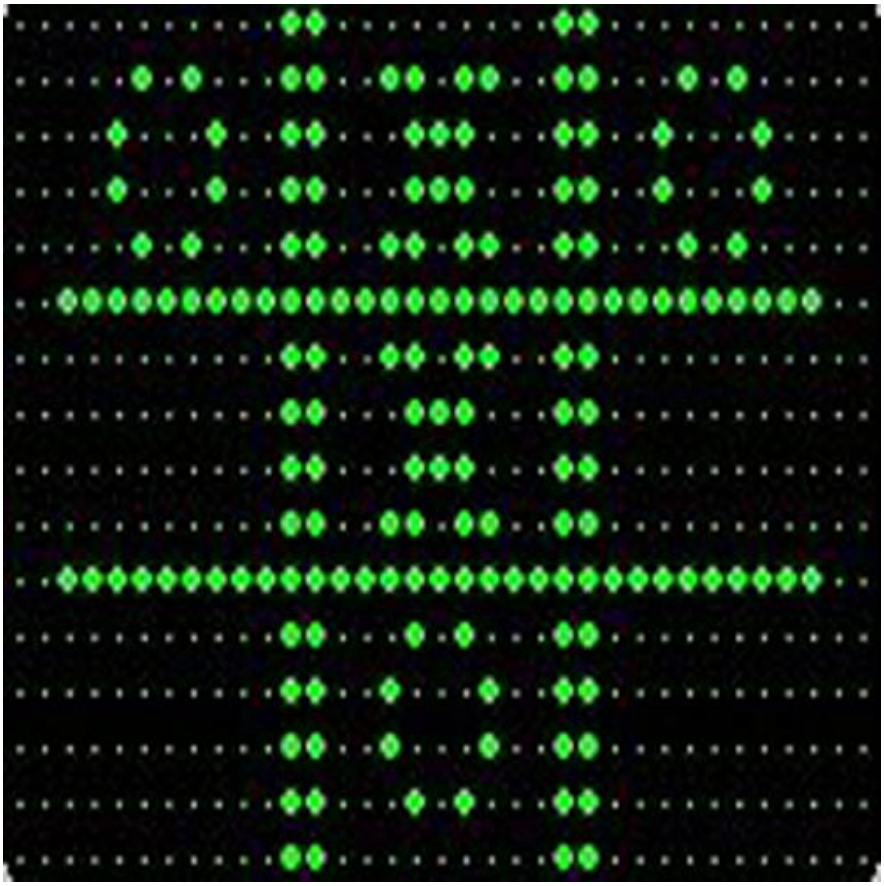
1.1 História dos jogos eletrônicos

Um dos primeiros indícios do início dos games nos microcomputadores vem de 1951 com Alexander Douglas, que era um doutorando da Universidade de Cambridge, Inglaterra. Ele adaptou o famoso “jogo da velha” para o computador da Universidade, chamando-o de OXO. O jogo era exibido em um tubo de raios catódicos de 35 x 15 pontos, como pode ser visto na **Figura 1.**

Dez anos mais tarde, em 1961, o Instituto de Tecnologia de Massachusetts, popularmente conhecido como MIT, recebeu o computador DEC PDP-1. Steve “Slug” Russell, junto com Marting “Shag” Graetz e Wayne Wiitanen, desenvolveu um jogo de batalha espacial baseado no seriado Lensman, chamando-o de Spacewar. O jogo consistia

em duas naves batalhando no espaço e foi espalhado pela comunidade universitária americana que estava nos primórdios do desenvolvimento de games para PC.

Figura 1 – Tela do OXO, o primeiro jogo que se tem notícia

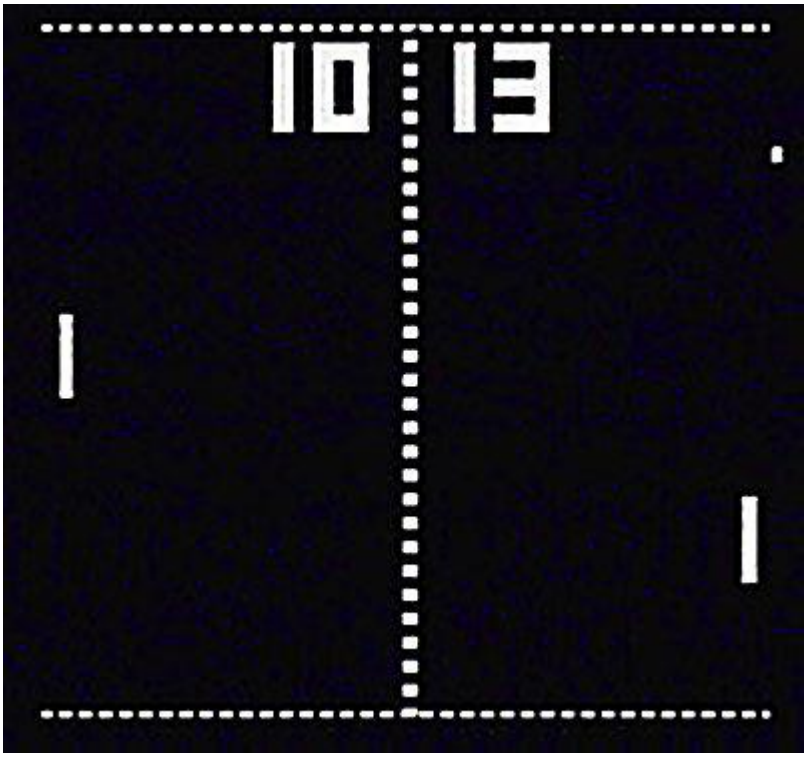


Fonte: Timeline of Computer History (1)

Em 1966, a primeira experiência de jogos na televisão foi feita com o engenheiro Ralph Baer. Seu primeiro projeto, chamado de Brown Box, permitiu que os jogadores pudessem jogar sem um computador, processador ou software. Já em 1972, o primeiro jogo que depois viria se tornar uma febre é lançado: Pong. Com gráficos simples, que podem ser visualizados na **Figura 2,** revolucionou os jogos arcade e popularizou aquilo que era só visto em universidades.

Atari lança seu primeiro videogame em 1977, o Atari 2600, usando um microprocessador 8-bit MOS 6507 e projetado para ser usado em conjunto com uma TV. Foi o primeiro grande sucesso de vendas de um videogame, mais de 20 milhões de unidades vendidas na década de 80.

Figura 2 – Tela do jogo Pong, o primeiro jogo eletrônico a se popularizar



Fonte: Timeline of Computer History (1)

A gigante Nintendo lançou seu primeiro console em 1983, o Nintendo Entertainment System, **Figura 3**. Foi responsável por apresentar ao planeta o personagem mais icônico da história do mundo gamer: Mario. Seis anos mais tarde, lançou o Game Boy, o primeiro videogame portátil amplamente comercializado pelo mundo e responsável por popularizar o jogo Tetris. Mais de 100 milhões de unidades vendidas em 20 anos.

A revolução 3D toma forma e em 1993 um dos jogos mais impactantes da história é lançado: Doom. Um jogo de tiro em primeira pessoa que revolucionou o mercado de jogos eletrônicos chamando atenção até do Governo dos Estados Unidos.

Ao longo do final da década de 90 e início dos anos 2000 os videogames e, consequentemente, os computadores se tornaram populares e expandiram os jogos para todas as partes do mundo.

As séries de consoles Playstation e Xbox levaram a níveis inimagináveis de popularização dos jogos eletrônicos. Além disso, hoje os smartphones representam uma grande parcela do mercado de games, aliados com a expansão dos jogos online em PC como Counter Strike e League of Legends. O mercado de jogos, em 2016, chegou a 99,6 bilhões de dólares movimentados segundo o relatório da NEWZOO (2). A influência

cultural está crescendo muito e eventos ligados a jogos eletrônicos reúnem milhares de pessoas como pode ser visto na **Figura 4**, em que o time brasileiro SK Gaming sagrou-se campeão em 2016 e embolsou 500 mil dólares.

Por outro lado, jogos eletrônicos foram um dos principais responsáveis pelo desenvolvimento de GPUs cada vez mais avançadas. Elas atualmente ajudam na solução de problemas reais, por exemplo, de biologia e criptografia com um alto poder de processamento, é o caso da tecnologia CUDA da Nvidia.

Figura 3 – O primeiro console da Nintendo (NES), responsável por clássicos como Mario e Zelda

Fonte: Timeline of Computer History (1)

Figura 4 – Arena em Colônia, Alemanha, lotada para a final da ESL One de Counter Strike Global Offensive

Fonte: ESL One (3)

2 DESCRIÇÃO DO PROJETO

A seção 2.1 deste capítulo tem como objetivo elucidar pontos importantes da implementação do jogo como variáveis e rotinas, a seção 2.2 irá relatar como foram tratados os elementos gráficos, por fim, a seção 2.3 explicará sobre como ocorreu a divisão de tarefas para a construção do jogo.

2.1 Principais rotinas e variáveis

O jogo “DodGG” parte do princípio de testar o reflexo do jogador e resistência. São definidos 3 níveis de dificuldade: fácil, médio e difícil. Cada nível com tempo de espera para movimentação da bomba diferente, ou seja, *delay* diferente, foi utilizada a função *delay* da biblioteca Irvine (4) para que isto fosse possível. A partir disso, o jogador tem um tempo limitado pelo *delay* para pressionar uma tecla de direção e mover o personagem, caso isto não ocorra, a bomba é movida. A tela do jogo é definida em 60 caracteres de largura e 20 de comprimento.

A movimentação da bomba corresponde a outra rotina importante desenvolvida.

Ela é responsável por sortear uma posição inicial pela função *RandomRange*, também da biblioteca Irvine (4). Sempre que uma nova bomba ou novo jogo é iniciado, a posição inicial é sorteada em um *range* de 1 a 48 (48 pois como a bomba tem 12 caracteres em uma linha, é preciso que ela seja desenhada dentro da tela definida de 60 caracteres). Suas movimentações para os lados e para baixo são sorteadas da mesma forma, em um *range* de 1 a 3, sendo cada número responsável por um movimento. As posições são sempre atualizadas em variáveis após uma movimentação, são elas "xBomba" e "yBomba". Ocorre também o controle de colisão com as laterais da tela, não permitindo que as bombas passem os limites estabelecidos.

A rotina da movimentação do personagem segue o que já foi dito no começo desta seção, caso o jogador aperte a tecla no espaço de tempo definido para o nível e a movimentação é ativada com uma chamada para o procedimento "moveperson" que atualiza os valores do eixo X. O personagem é fixado no final da tela, em Y = 15, podendo se movimentar para esquerda e direita, não ultrapassando os limites laterais. As posições são atualizadas em uma variável: "xPerson". Existe também "yPerson", porém ela é uma constante com valor 15.

A rotina mais importante é a de checagem de colisão entre personagem e bomba, ela que define se o player irá pontuar ou não. A decisão é sempre chamada quando a última linha da bomba, o seja "yBomba" atinge Y = 15, que é onde se encontra a cabeça do personagem. Chegando nesta posição, outras rotinas denominadas como "checaconflito", "checax1person" e "checax2person". Estas serão responsáveis por comparar a posição no eixo X entre a bomba e o personagem. Se estiver dentro do intervalo do personagem, é chamada a rotina "explodiu", caso contrário é chamada a rotina para criação de uma nova bomba "novabomba".

A rotina de explosão é uma chamada para que seja desenhada a explosão e posteriormente a chamada de game over com a rotina "go", que escreve na tela "GAME OVER" e a pontuação final obtida pelo usuário.

Já a rotina "novabomba" é mais complexa. Primeiramente é feita a contagem de pontos incrementando a quantidade de bombas que já desceram, armazenada na variável "qtdObj", e multiplicando pelo nível definido pelo nível selecionado (1 para fácil, 2 para médio e 3 para difícil, como pode ser visto na **Equação 1**. Diferentemente do que foi definido em relatórios anteriores, a equação para cálculo da pontuação foi simplificada,

de modo que facilitasse a implementação e visualização do usuário. O placar é armazenado na variável "placar". Uma nova bomba é gerada através da função *RandomRange* e suas posições são atualizadas nas variáveis "xBomba" e "yBomba".

$$placar = levelSel * qtdObj \quad (1)$$

As variáveis utilizadas no projeto são descritas na **Tabela 1,** com seus respectivos nomes, tipos e bytes totais utilizados.

Tabela 1 – Variáveis utilizadas no projeto

Nome	Tipo	Tamanho total (bytes)
Ti (tela inicial)	BYTE	741
Ajuda	BYTE	741
Personagem	BYTE	36
Bomba	BYTE	60
Explosao	BYTE	374
Gameover	BYTE	600
Totalpontuacao	BYTE	48
StrPlacar	BYTE	12
Espaco	BYTE	2
Linhanula	BYTE	66
Levelsel	BYTE	1
XBomba	BYTE	1
YBomba	BYTE	1
XPerson	BYTE	1
yPerson	BYTE	1
Placar	DWORD	4
QtdObj	DWORD	4

Fonte: Elaborado pelos autores.

2.2 Tratamento dos elementos gráficos

Outra mudança em relação à relatórios anteriores é o tamanho da tela de jogo, que passou de 60 x 30 caracteres para 60 x 20. Os componentes gráficos são os mesmos:

- Bomba – 12 x 5 caracteres
- Personagem – 6 x 6 caracteres

Uma parte não menos importante do jogo é a solução do problema de *flickering*, que consiste na tela piscar sempre que era apagada. A solução encontrada passa pelo uso de outra função da biblioteca Irvine (4), denominada *goToXY*. Sempre que uma nova posição da bomba será desenhada, a função é chamada para o Y anterior da bomba e move o cursor para esta linha. Assim, os "rastros" deixados pelo desenho são apagados com a inserção de linhas nulas, somente com espaços. Após este procedimento, o cursor é

movido para a posição da bomba atual, para que possa ser desenhada.

O mesmo ocorre com o desenho do personagem: sempre que ocorre uma movimentação, linhas nulas são adicionadas a partir de “yPerson” e depois o cursor volta para a posição desejada do desenho.

2.3 Divisão de tarefas

O aluno Felipe Chinen foi responsável por construir os gráficos iniciais do jogo, como a tela inicial, a de explosão e a de fim de jogo. Além disso, construiu as rotinas de checagem de colisão entre personagem e bomba.

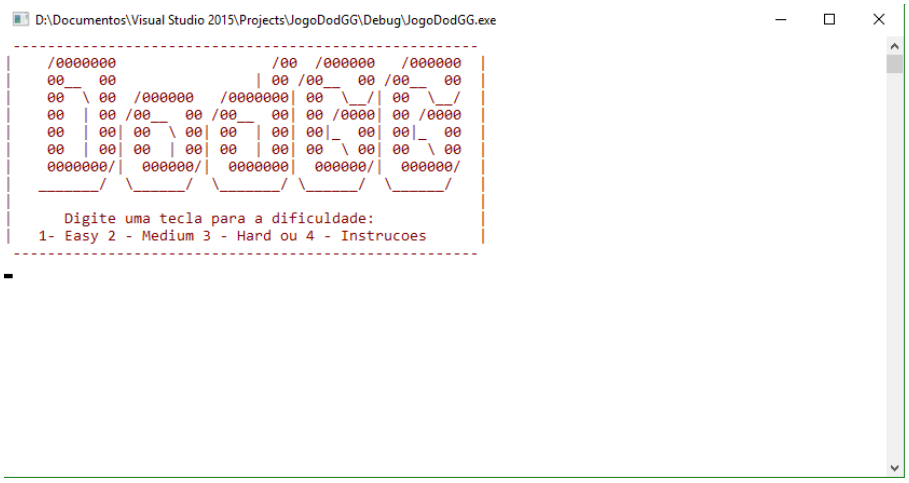
Já o aluno Igor fez as rotinas de tratamento dos elementos gráficos e correção de *flickering*. Também fez as rotinas de pontuação e, consequentemente, inserção de novas bombas.

As rotinas de movimentação da bomba e do personagem foram feitas em parceria, devido à dificuldade encontrada por se relacionar tanto com rotinas feitas por Felipe quanto por Igor.

3 ELEMENTOS ESPECÍFICOS

A primeira funcionalidade vista pelo usuário é a seleção do nível de jogo ou então a escolha para ver as instruções antes de jogar. A tela inicial pode ser visualizada pela **Figura 5**. Ao apertar 1, 2 ou 3 o jogador irá para a tela de jogo, caso aperte 4 será levado para a tela de ajuda, podendo voltar para o início depois. A explicação pode ser acompanhada com o diagrama de estados presente na **Figura 6.**

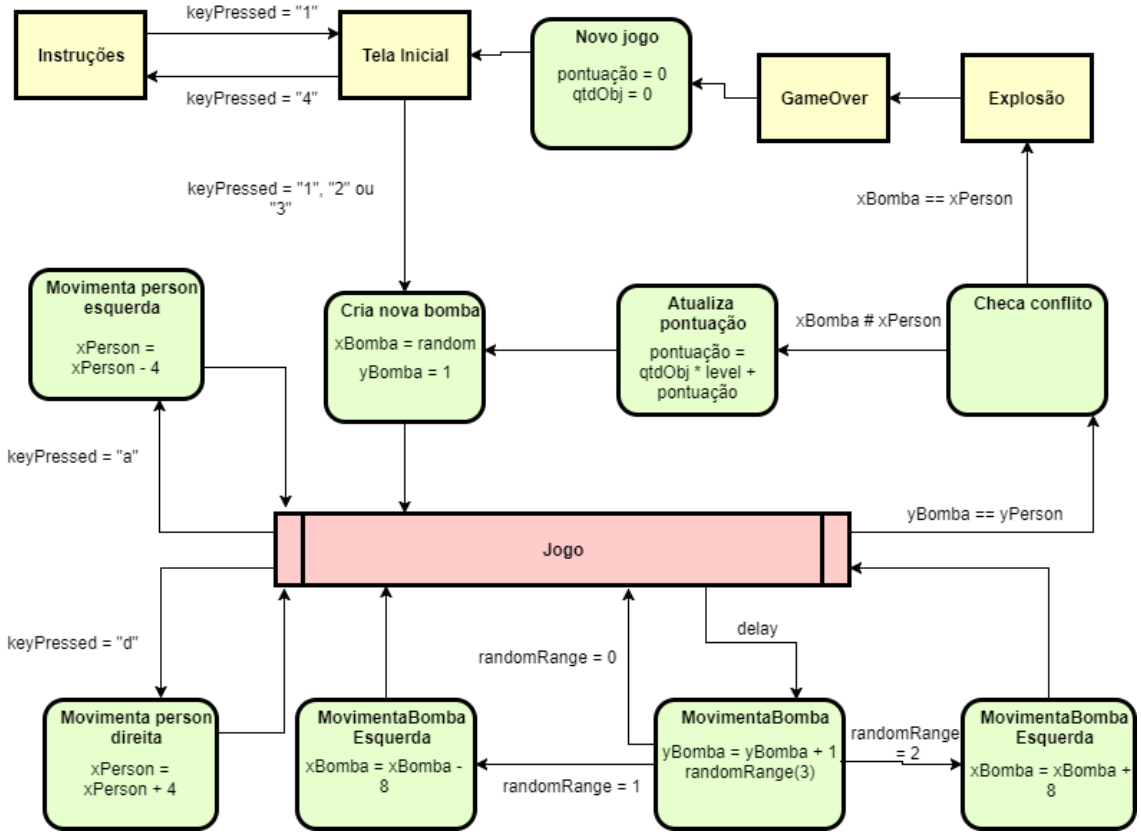
Figura 5 – Tela inicial



Fonte: Elaborado pelos autores.

A tela de ajuda pode ser visualizada na **Figura 7**, em que são mostradas as opções de movimentação para o personagem e as instruções para que isto ocorra devidamente. As teclas definidas para movimentação são “a” e “d”, não podendo ser maiúsculas pois têm códigos ASCII diferentes. Além destas instruções de movimentação, são mostradas as instruções para seleção de nível. Para voltar à tela inicial é necessário apertar 1.

Figura 6 – Diagrama de estados do jogo

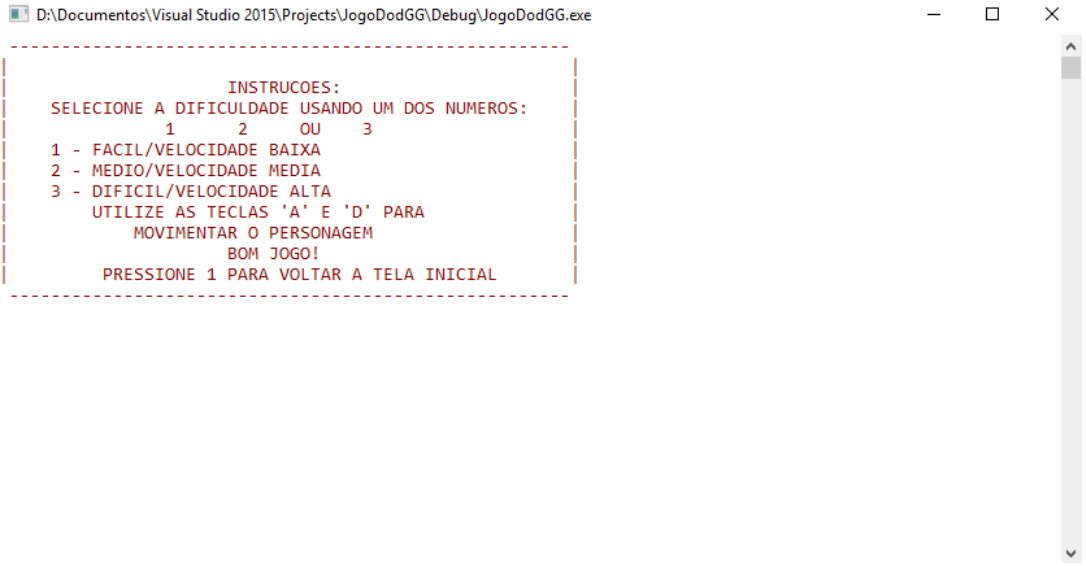


Fonte: Elaborado pelos autores.

A tela de jogo é composta por uma linha onde está a pontuação e o resto das linhas pela bomba e personagem, como pode ser visualizado na **Figura 8**. A movimentação da bomba, como já descrita neste relatório, ocorre de forma aleatória e após o período de tempo pré-determinado pelo nível. Já a do personagem depende da tecla pressionada pelo usuário.

As outras telas do jogo, explosão e game over, são simplesmente desenhos fixos sem funcionalidades, como pode ser visto nas **Figuras 9** e **10** e representadas no diagrama de estados. Após estas duas telas, o jogador é levado de volta para a tela inicial e pode escolher outra ou a mesma dificuldade para jogar.

Figura 7 – Tela de ajuda

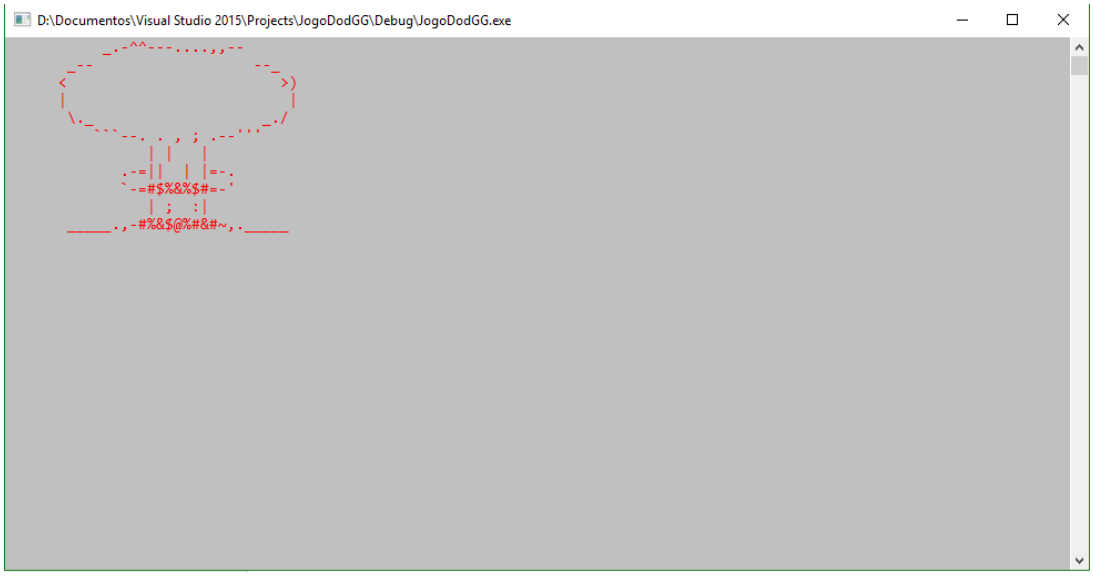


Fonte: Elaborado pelos autores.

Figura 8 – Tela de jogo

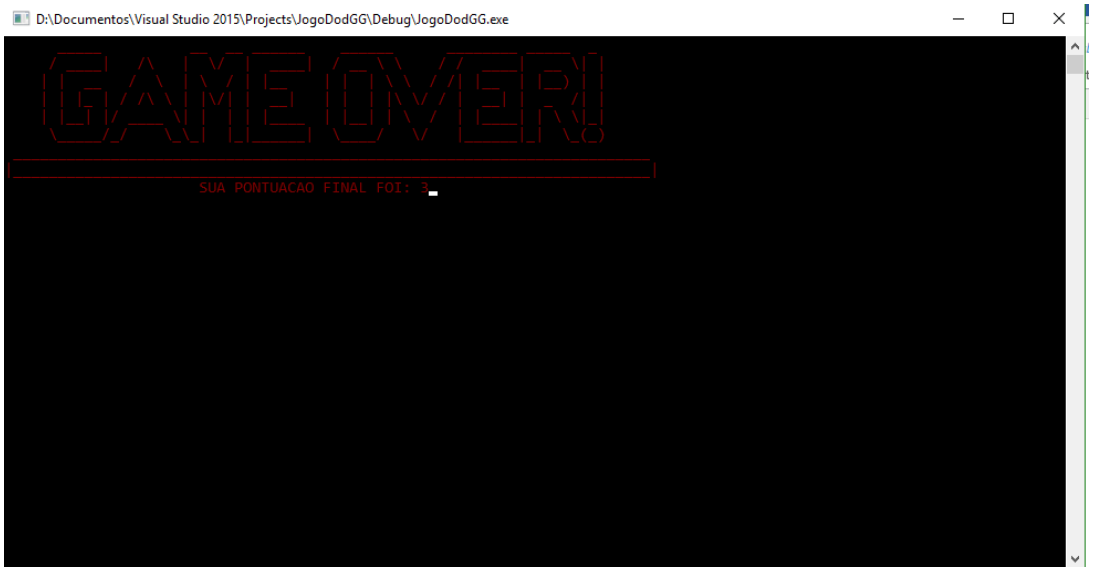
Fonte: Elaborado pelos autores.

Figura 9 – Tela de explosão



Fonte: Elaborado pelos autores.

Figura 10 – Tela de game over



Fonte: Elaborado pelos autores.

4 ANÁLISE CRÍTICA E DISCUSSÃO

O projeto teve como objetivo aprofundar conhecimentos na linguagem Assembly para a arquitetura x86. O planejamento inicial foi completado com êxito, o desenvolvimento do projeto, desde o começo, seguiu o que foi proposto nos relatórios anteriores, mudando apenas a fórmula de pontuação para facilitar a compreensão dos jogadores e o tamanho da tela de jogo, para trazer mais dificuldade ao game.

Uma dificuldade grande encontrada foi a de solucionar o problema de *flickering*.

A ideia inicial de limpar toda a tela usando a função *clrscrn* da biblioteca Irvine (4) foi mudada para o apresentado na seção 2.2 deste relatório. De fato, a solução encontrada é mais elaborada e foi o que mais demandou tempo de implementação. Algumas vezes o que era esperado saía de controle apenas por um valor errado em um registrador e necessitava que a implementação fosse revista para corrigir *bugs*.

Outro ponto importante da implementação, foi a dificuldade de alinhar *jumps* e rotinas. O código não foi otimizado por falta de tempo e por isso a utilização de *jumps* condicionais foi utilizada com exaustão, o que com certeza não leva a uma solução ótima em desempenho. Por fim, foi possível explorar a biblioteca Irvine (4) e descobrir funções muito úteis para o projeto, como a *gotoxy* e *delay*.

"DodGG" partiu do princípio de ser um jogo simples que tentasse testar os reflexos e resistência de seus jogadores. O grupo acredita que isto foi alcançado e também que os conhecimentos na linguagem foram aprofundados, em grande parte, por causa do desenvolvimento deste game.

REFERÊNCIAS BIBLIOGRÁFICAS

1 HISTORY, T. of C. 2017. Disponível em: http://www.computerhistory.org/timeline/graphics-games/. Acesso em: 06 de jul. 2017.

2 NEWZOO. 2016 Global games market report. Disponível em: https://cdn2.hubspot.net/hubfs/700740/Reports/Newzoo_Free_2016_Global_Games_Market_Report.pdf. Acesso em: 15 de abr. 2017.

3 ESL One Cologne. 2016. Disponível em: http://en.esl-one.com/csgo/cologne-2016/. Acesso em: 06 de jul. 2017.

4 IRVINE, K. R. *Irvine Library Help.* [S.I.]. Disponível em: http://programming.msjc.edu/asm/help/. Acesso em: 03/06/2017.