CENTRO UNIVERSITÁRIO UNIFACVEST
CURSO DE CIÊNCIA DA COMPUTAÇÃO
TIAGO SILVA MORAES

RANCE WORD'S END I (VERSÃO 1.2.25)

LAGES
2016

TIAGO SILVA MORAES

RANCE WORD'S END I (VERSAO 1.2.25)

Monografia apresentada à Centro Universitário Unifacvest – FACVEST como requisito para obtenção do título de Bacharel em Ciência da Computação.

Email:tiagohitmansilvamoraes@gmail.com

Orientador: João Francisco Frank Gil

LAGES

2016

TIAGO SILVA MORAES

RANCE WORD'S END I (VERSAO 1.2.25)

Trabalho de Conclusão de Curso de Ciência da Computação apresentada à Centro Universitário Unifacvest – FACVEST como requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientador: João Francisco Frank Gil

Lages, SC ___/___/2016.

Nota _____

_____

Márcio José Sembay

Coordenador do curso de graduação

LAGES

2016

RESUMO

O presente trabalho tem como objetivo apresentar a colaboração encontrada na realização de projetos, como aspecto relevante à Educação, o curso teve como premissa a construção de Rpg eletrônicos como contexto pedagógico e, nesse sentido, grande parte dos projetos executados para a construção deste tipo de jogo, assim, tomamos as discussões registradas no ambiente do curso, o projeto, a orientações feitas a partir deste projeto e o produto final desse processo, como dados de análise para as considerações desse descreve a utilização do Software em uma ação interdisciplinar de trabalho. A proposta de atividade era o planejamento e o desenvolvimento de um jogo eletrônico. O Rpg Maker VX Ace mostrou-se um software que permite a aprendizagem e que oferece recursos que possibilitam desenvolver jogos diversificados e interessantes. A experiência foi proveitosa na abordagem de mídia-educação. As relações entre software e aprendizagem estão, em especial, diretamente condicionadas à metodologia de ensino utilizada.

Palavras chaves: Jogos, RPG, Aprendizagem, Dados.

ABSTRACT

This study aims to present the collaboration found in the realization of projects, as relevant aspect to education, the course was premised on the construction of electronic Rpg as a pedagogical context and in that sense, most of the projects carried out for the construction of this type of game, so we take the discussions recorded in the course environment, the project, the guidelines made from this project and the final product of this process, and analysis of data to the considerations that describes the use of the Software in an interdisciplinary action work. The proposed activity was the planning and development of an electronic game. The RPG Maker VX Ace proved a software that allows learning and offers features that enable developing varied and interesting games. The experience was fruitful in media-education approach. The relationship between software and learning are, in particular, directly conditional on teaching methodology.

Key words: Games, RPG, Learning, Data.

RESUMEN

Este estudio tiene como objetivo presentar la colaboración que se encuentra en la realización de proyectos, como aspecto relevante a la educación, el curso se basa en la construcción de RPG electrónico como un contexto pedagógico y en ese sentido, la mayoría de los proyectos llevados a cabo para la construcción de este tipo de juego, así que tomamos las conversaciones grabadas en el entorno del curso, el proyecto, las directrices elaborados a partir de este proyecto y el producto final de este proceso, y el análisis de los datos de las consideraciones que describe el uso del Software en una obra de acción interdisciplinaria. La actividad propuesta fue la planificación y el desarrollo de un juego electrónico. El RPG Maker VX Ace demostró un software que permite el aprendizaje y ofrece características que permiten el desarrollo de juegos variados e interesantes. La experiencia fue fructífera en el enfoque de los medios de comunicación-educación. La relación entre el software y el aprendizaje son, en particular, directamente condicionada a la metodología de enseñanza.

Palabras clave: Juegos, juegos de rol, de aprendizaje, de datos.

LITA DE FIGURAS

LISTA DE SIGLAS

A.P.Is	-	Application Programming Interface
C.R.P.G	-	Computer Role Playing Game
D.K	-	Donkey Kong
E.U.A	-	United States of America
I.A	-	Inteligência Artificial
Inde.	-	Independente
OpenAL	-	Open Audio Library
OpenGL	-	Open Graphics Library
R.G.S.S 3	-	Ruby Game Scripting System
R.P.G	-	Role Playing Game
T.I	-	Tecnologia da Informação
N.A.S.S.A	-	North American Shetland Sheepbreeders Assoc.

Sumário

1. INTRODUÇÃO

No desenvolver o *programa* é uma atividade recente na história, se comparada com ciências existentes, ainda estamos aprendendo como tudo isso funciona e a melhor forma de lidar com o software.

As principais empresas de games, suas crises e suas gerações dos jogos bem como seus criadores, Ralph Bear, Shigeru Miyamoto, e seus jogos: Mário bros, Mário bros. Lost elevens, Mario bros. 3, new super Mario, Mario rpg, the legend of zelda, portal, autor (SOUZA e ROCHA, 2005) , no autor (BATISTA, QUINTÃO & CAMPOS**,** 2007).

O *Role Playing Game*, conhecido pela sigla RPG, surgiu nos EUA na década de 70 como um jogo de mesa, em que, conforme a tradução do próprio nome sugere, baseia-se na interpretação de papéis.

Nos tempos em que não havia internet, na época de disquetes e fitas de vídeo game, o porem nos dias atuais os jogos evolucionaram tanto em que fitas e CD's ficaram ultrapassados com a nova era do Blu-ray, em que, tem maior capacidade de armazenamento que vai até uma capacidade de espaço definição de como a ferramenta deverá ser usada?

1.1. JUSTIFICATIVA

Em fazer um jogo, ao entendimento de um processo de um jogo Inde. (*Independentes*) e que pode também não ser bem-sucedida, apresenta conceitos históricos e conceitos que poderão ser usados em criação de games, para quem goste de RPG.

Com base no documentário da era dos vídeos games, será adaptado a ferramenta do jogo a uma aplicação educacional como desafio intelecto ou outra área em questão, no uso RPG Maker VX ACE na linguagem RGSS3 (*Ruby Game Scripting System*) para o aprendizado da linguagem inglesa, espanhola.

1.2. **IMPORTÂNCIA**

O engenheiro Ralph Bear teve a brilhante ideia de um aparelho interativo, construiu do zero em Loral em 1951 no Bronx, Nova Iorque. Bear criou um videogame simples de dois jogadores em que podia ser exibido em uma televisão padrão chamado chase, em que dois pontos seguir um a outro na tela.

No editor do rpg Maker, poderá criar mapas utilizando tilesets conhecidos também como "chipsets", o nome *tileset* veio a ser usado a partir do RPG Maker XP para frete, programar acontecimentos no jogo através de comandos de eventos pré-programados, entre outras funções, no entanto, em base educativa.

1.3. **OBJETIVOS GERAIS**

Tem como objetivo desenvolver um jogo de R.P.G (*Role Playing Game*) independente com a aprendizagem na linguagem RGSS3 (*Ruby Game Scripting System*) e na resolução de enigmas e sobre a história da primeira guerra mundial.

1.3.1. **OBJETIVOS ESPECÍFICOS**

Para que o jogo obtenha sucesso para o aprendizado do Jogo chegando as necessidades do usuário principal objetivo deste trabalho, sendo assim, que projeto seja com mais facilidade de entendimento no seu andamento. Abaixo serão listadas as funcionalidades desenvolvidas neste projeto:

a) Usando a linguagem RGSS3 para desenvolver a ferramenta do jogo.

b) Implementar o jogo contará com relatos históricos uma das grandes guerras.

c) Colocação de rotinas de I.A (*Inteligência Artificial*) no jogo.

Na sessão seguinte serão explicadas de forma ampla e claramente o rpg Maker:

2. RPG MAKER

1.1. DESENVOLVIMENTO DE UM JOGO 2D E 3D

Ao desenvolvimento de um jogo em 2D existem vários tipos de compiladores com Java, Dark Basic, Boo, C#, C, Pascal, RGSS3, dentre outros em que podem desenvolver e criar o seu jogo em 2D, hardwares ”Processadores” eficientes da Intel (Corel i5/i3/i7) também da AMD. (**ALVES**, 2007).

Em uma plataforma em 3D destaca-se em na modelagem, na jogabilidade, concepção de arte, jogos de luta como mortal kombat de 1992 adota-se um modelo de digitalização de movimentos de pessoas animação de atores e seus movimentos. **(WALTER, CLUA, & BITTENCOURT**, 2004).

1.2. HISTORIA DO RPG

O *Role Playing Game*, conhecido pela sigla RPG, surgiu nos EUA na década de 70 como um jogo de mesa, em que, conforme a tradução do próprio nome sugere, baseia-se na interpretação de papéis.

Os jogadores se reúnem em volta de “uma mesa” e de forma descritiva, e às vezes até teatral, vivenciam uma aventura fantasiosa. Um deles assume o papel de mestre, que atua como o narrador, roteirista e árbitro da história (**BITTENCOURT; GIRAFFA,** 2003). Durante uma partida, o mestre conta uma história criada por ele e insere nela os personagens criados por cada um dos demais jogadores.

A cada momento, descreve as situações que os jogadores estão enfrentando e pede que eles descrevam o que os seus personagens fariam diante delas. Então, o mestre precisa dizer o resultado da ação de cada personagem. É comum o uso de dados poliedros cujas faces são numeradas de um até o número total de faces, que introduzem aleatoriedade à narrativa e ajudam a não haver poder demais concentrado nas mãos do mestre.

Este também tem a função de interpretar personagens secundários que surjam no decorrer da trama. Sendo assim, possível notar que no RPG de mesa não há uma imposição de regras rígidas como nos jogos de computador tradicionais. Já que o árbitro do jogo é um ser humano e não uma máquina, o RPG ajuda a atacar um dos grandes desafios no desenvolvimento dos jogos educacionais, que é a união entre a imaginação do jogador e o jogo. (**MAIKE; MIRANDA; BARANAUSKAS**, 2011)

1.3. UTILIZAÇÃO DO RPG MAKER VX ACE E CONSOLES

As ferramentas utilizadas para fazer tanto 3D como 2D são elas Unity, RPG Maker, Mugen, C++, Java, C, Pascal, Dark basic, Boo, RGSS3 o código a seguir mostra a dificuldade em que o jogador poderá escolher no jogo de um custo de 123,99 reais, vai ter o uso de adobe photoshop CS4 para fazer algumas mudanças.

No ano de 1997 e lançado no Japão pela impressa ASCII, porém, não teve muito sucesso pois sua limitação nos gráficos do programa, então foi substituída por RGSS a versão final da ferramenta do jogo. (**ALVES**, 2007**;NASCIMENTO;ALCÂNTARA,** 2014).

O argumento em que jogos podem ter sim um tipo de cultura e que podem também a motivação de crianças, jovens e adultos ao processo de aprendizagem (**GEE,**2009**; TAVARES,**2012**, SANTOS,** 2014**).** Exemplo das lógicas do programa e parte de um código em RGSS3 no apêndice A e B.

1.4. DIFERENÇAS DE MÉTODOS PIAGET E VYGOTSKY

Métodos entre dois pensadores em que um tipo de ideologia em que o ser vivo do seu desenvolvimento, biológico e social, em que, evolução mental de acordo com os períodos de idade, no entanto o progresso humano se dá pelo processo relacional, já que conhecimentos elaborados de acordo com os estágios de andamento: sensório-motor, pré-operacional, operações concretas e operações formais.

No qual a inter-relação dos fatores internos e externos responde pelos processos mentais no fator de aprendizagem eles tem uma ideia pedagógica diferente, porém estão falando em termos educacionais em que a criança possa aprender com mais facilidade no ensino ao decorrer de seu tempo. (**SOUZA, KRAMER,** 1991)

1.5. COMPUTADORES

A era dos computadores estava começando com uma jogada sombria começa os jogos R.P.G (*Role Playing Game*) os mais conhecidos Mistery House, no qual, o objetivo do jogador é investigar uma casa abandonada e o que estaria acontecendo, outro conhecido pelos jogadores é o Tetris um dos melhores de sua simplicidade ao divertimento do sujeito, um game famoso também Test Drive de 1987 um percursor ao gênero de corrida. (**ALVES, MARSAL; PEREIRA, 2006; SARTINI & GARBUGIO, 2004; WALTER ET AL.,** 2004)

2. INTERFACE

O caráter de um jogo não está entre gráficos pois a importância dos gráficos dentro dos vídeos games em que sua organização cronológica é completamente diferente de outra, nesta diferença em que a momentos que interface é comprometida no mercado de trabalho (**PINHEIRO & BRANCO**, 2006).

3. MÍDIAS DIGITAIS VESUS MÍDIAS FÍSICAS

Para um lado e outro a mídia digital tem suas vantagens pelo fato em que o preço é mais acessível, porém usa várias contas para usá-las, já na física possível alto preço do produto, mas em compensação, não necessita de contas extras no vídeo game. (**CLUA**, 2004).

3.1. MÍDIA DIGITAL

A mídia digital tem a vantagem de não corromper seus dados e ter a disposição do cliente, as desvantagens sobressair contas em contas, baixar os downloads, a formatação do vídeo game ps3 e tiver um jogo em mídia digital perde o jogo pôr ele poderá baixar novamente, no entanto

se usuário perdeu sua conta e a conta antiga estiver guardada em um jogo este game será perdido por completo. (**WEILLER**, 2010; **LUNA**, 2012)

3.2. MÍDIA FÍSICA

As mídias físicas mais antigas estão se tornando mais difíceis de conseguir aos dias atuais, o caso de fitas raras do snes conhecido como Super Nintendo aqui no ocidente (Super Famicon) no oriente fitas que custam muito caro em sites custa mais R$ 1000,00; pois eram de competições que faziam na época e que hoje tem poucas rodando no mercado até então. (**WEILLER**, 2010)

4. JOGOS DIAS ANTIGOS

Nos tempos em que não havia internet, na época de disquetes e fitas de vídeo game, o porém nos dias atuais os jogos evolucionaram tanto em que fitas e CD's ficaram ultrapassados com a nova era do Blu-ray, em que, tem maior capacidade de armazenamento que vai até 500gb de espaço definição de jogo Exercício ou passatempo entre duas ou mais pessoas das quais uma ganha, e a outra, ou as outras, perdem. O que serve para jogar determinado. (**BATISTA, QUINTÃO & CAMPOS,** 2007)

4.1. INÍCIO DOS JOGOS E OS PIONEIROS

No começo de guerra fria surgiu o primeiro jogo de entretenimento em um Osciloscópio de 1958, conhecido como tênis para dois e processado em um computador analógico da N.A.S.S.A (*North American Shetland Sheepbreeders Assoc.)* nos Estados Unidos (**BATISTA, QUINTÃO & CAMPOS,** 2007).

4.2. O PRIMEIRO CONSOLE E 1ª GERAÇÃO

O engenheiro Ralph Bear teve a brilhante ideia de um aparelho interativo, construiu do zero em Loral em 1951 no Bronx, nova Iorque. Bear criou um videogame simples de dois

jogadores em que podia ser exibido em uma televisão padrão chamado chase, em que dois pontos seguir um a outro na tela.

Um protótipo foi demonstrado em fevereiro de 1968 para vice-presidente da teleprompter, Hubert Schlafly, no que, assinou um acordo com Sanders, no desenvolvimento do hardware e jogos continuou resultados no protótipo final "A caixa marrom", que tinha dois controles, uma pistola de luz e seis interruptores no console que selecionavam o jogo. Ele abortou vários fabricantes de televisão nos E.U.A e um acordo assinado com Magnavox no final de 1969(**SOUZA e ROCHA**, 2005).

4.3. GERAÇÃO 8 BITS

Em 1985 e criado o nes conhecido como nintendinho ou Famicon no Japão, foi bem atrativo que a sega coloca no mercado um ano antes de seu rival o máster system de 1986, só na Europa que ganhou desta que, pois a falta de Computer Role Playing Game que deixou a hegemonia do NES. (**WALTER** , 2004)

4.4. GERAÇÃO 16 BITS

No ano de 1988 a sega lança seu console Genesis tentando derrubar a concorrente. No Japão os jogos CRPG (Computer Role Playing Game) invadem e acertam o interesse do público. Depois de dois anos, a Nintendo lança seu console o Super Famicon no Japão e nos EUA Super Nintendo em 1990, apostou nas novidades e renovação. Mario garantia uma franquia milionária para empresa que se solidifica até os dias atuais. (**WALTER**, 2004; **LUTERMAN**, 2012)

4.5. GERAÇÃO 32 BITS

Começando a falar dos maiores fracassos de 1993, o console 3DO Trip Hawkins o criador da Eletronic Arts. Mas em 1994 a Sony resolve entrar no mercado dos vídeos games que ela mesma produziria, o console chamado de Playstation. No mesmo ano a Sega lança o Sega saturno, porém ele não teve muito sucesso em vendas com a dificuldade de processar em assemble. (**WALTER** , 2004)

4.6. GERAÇÃO 64 BITS

Nesta geração de 64bits ao fracasso do console Atari Jaguar de 1993, seu rival tinha um novo console chamado de Nintendo 64 de 1996 com as franquias Pokémon, Super Mario 64, Mário Kart 64, Donkey Kong 64 entre outros, conquistou um grande público pelos gráficos melhorados, pois de textura muito repetitivas e cartuchos muito caros, porém em original ao acesso de títulos antecessores dos seus sucessos (**WALTER ET AL.**, 2004**;SOHN**, **HELOU**, **STEIL**,2012).

4.7. GERAÇÃO 128 BITS

No ano de 1998 lança o Dreamcast da Sega, o console teve um curto período no mercado por apenas dois anos, no entanto a sega com dividas até o pescoço, então, decidiu fazem programas de software. No século XXI em 2001 a Microsoft entra no mercado de games com o novo console o Xbox, usando a tecnologia Open GL no console e nos computadores o DirectX. Microsoft lança seu novo modelo de console o Xbox 360. (**ALVES, MARSAL; PEREIRA, 2006; SARTINI & GARBUGIO, 2004; WALTER ET AL.,** 2004)

5. HISTORIA DO RPG MAKER

Enterbrain e ASCII lançaram o RPG Maker 2000 no Japão em 1999. Ele foi traduzido para o inglês e ilegalmente distribuído por um programador russo apelidado de Don Miguel, que também participou na tradução do RPG Maker 95 e de vários outros programas da série Maker da ASCII/Enterbrain.

Em 2002, Don Miguel recebe um aviso da Enterbrain. Consequentemente, no entanto, descontinuo seus trabalhos relacionados ao RPG Maker 2000, removeu o programa do site e incentivou os usuários do programa a comprarem uma versão legal do programa (em japonês). Devido à popularidade do programa, e a falta de uma tradução legal, a distribuição ilegal do programa continuou. Versões traduzidas do RPG Maker 2003 e XP foram distribuídas em todo

o mundo. Ainda hoje, grande parte dos utilizadores destas versões acreditam que o RPG Maker é gratuito, pois assim foi declarado quando disponibilizado pelos primeiros tradutores.

O RPG Maker permite que os usuários criem seus próprios jogos de RPG e com algumas mudanças no sistema pode criar até outros tipos. São similares aos populares jogos para Super Nintendo, a não ser que a imaginação do quem o utiliza, vá além dos recursos possíveis, criando efeitos inacreditáveis e surpreendentes, usando apenas o sistema permitido de 256 cores. No editor, poderá criar mapas utilizando tilesets conhecidos também como "chipsets", o nome *tileset* veio a ser usado a partir do RPG Maker XP, programar acontecimentos no jogo através de comandos de eventos pré-programados, entre outras funções.

Na versão XP apresenta também, um script embutido, baseada na famosa Ruby, o Ruby Game Scripting System (RGSS). Todas as versões do programa incluem o RTP, um pacote de gráficos e sons comuns entre todos que usam o RPG Maker (não irá rodar a não ser que você instale antes o seu respectivo RTP) prontos para serem utilizados nos jogos. (**MAIKE; MIRANDA; BARANAUSKAS**, 2011)

6. METODOLOGIA

6.1. DOCUMENTAÇÃO

A documentação bibliográfico cientificas, em que, mostra o avanço tecnológico de máquinas de entretenimento desde surgiu o primeiro jogo nos Estados Unidos em uma base militar por um osciloscópio, até as novas gerações que possuem mídia digital ou a física de Blu-ray, do surgimento de ferramentas no auxílio de criar jogos com temáticas distintas segundo (MENDES, 2011).

6.2. NATUREZA DA PESQUISA

O trabalho presente apresenta informações por meio de pesquisas de tipo teóricos, o objetivo da pesquisa teórico, artigos e revistas cientificas métodos eficientes na intenção de incentivar usuários com jogos educativos.

6.3. TIPO DA PESQUISA

O autor (WALTER ET AL., 2004) de método qualitativo de modo que modelo em 3D de hoje não se comparar os de antes de modo que os jogos mais antigos tenham uma capacidade gráfica inferior do que os dias atuais que o jogo mortal kombat XI foi um sucesso de vendas e a jogabilidade conceito de um produto ou um software seja de sucesso em 3D ou 2D e a qualidade de gráficos e a qualidade na jogabilidade.

6.4. TÉCNICAS DE PESQUISA

Pesquisa bibliográfica abrange toda a bibliografia disponibilizada que esteja relacionada ao tema de estudo. Abrange desde publicações, livros, monografias, pesquisas, teses, etc. Pesquisa de teórica, é a utilizar com o objetivo de conseguir informações e conhecimentos sobre um determinado problema, utilizando a observação referentes ao tema. (WALTER ET AL., 2004)

6.5. VALIDAÇÃO DO PROJETO

Buscando validar a ideia e a viabilidade do projeto para o público-alvo escolhido, foi realizada uma análise, através série de pesquisas. Pesquisando em *desenvolvedores* de *software* de todo o Brasil em que:

a. Métodos de desenvolvimento precisos ao usuário.

b. Visualização simples com dificuldade mediana

7. PROJETO

7.1. HARDWARE

No fato de todas as tecnologias que foram utilizadas serem usadas em um computador, o desenvolvimento ocorre em um estágio. O primeiro estágio do desenvolvimento foi a concepção do projeto, os primeiros passos. Essa fase foi desenvolvida e hospedada em uma máquina pessoal, com sistema operacional *Windows 10*, processador *Intel Core 2 Dou*, 4GB de memória RAM e capacidade de armazenamento de 500GB. Tendo em vista que a aplicação não está sendo utilizada por vários usuários, na sua execução irá ocupar apenas pequena parcela do *hardware* utilizado.

7.2. CRONOGRAMA DE ATIVIDADE

No cronograma de atividade mostra o período de tempo de desenvolvimento do projeto:

MESES / ATIVIDADES	SET				OUT				NOV				DEZ			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Adicionar elementos históricos	x															
Eventos	x	x	x	x												
Desenvolvimento no Jogo					x	x	x	x								
Correções								x	x	x	x	x				

7.3. TELAS

7.3.1. TELA DE INICIO

Demonstra o início da tela inicial do jogo, em que foi modificada nas messes do desenvolvimento do projeto.

Figura 1 Tela de inicio

6.3.2. DIFICULDADE

Insere no jogo um sistema de dificuldade, onde a escolha de certo nível resultará em uma alteração nos atributos dos inimigos, tornando-os mais fortes ou mais fracos. Uma das quatro opções só será liberada para o jogador se ele já tiver terminado o jogo pelo menos uma vez, apresentado no apêndice A.

Figura 2 Dificuldade do jogo

6.3.3. HISTÓRIA

Representação da história do jogo.

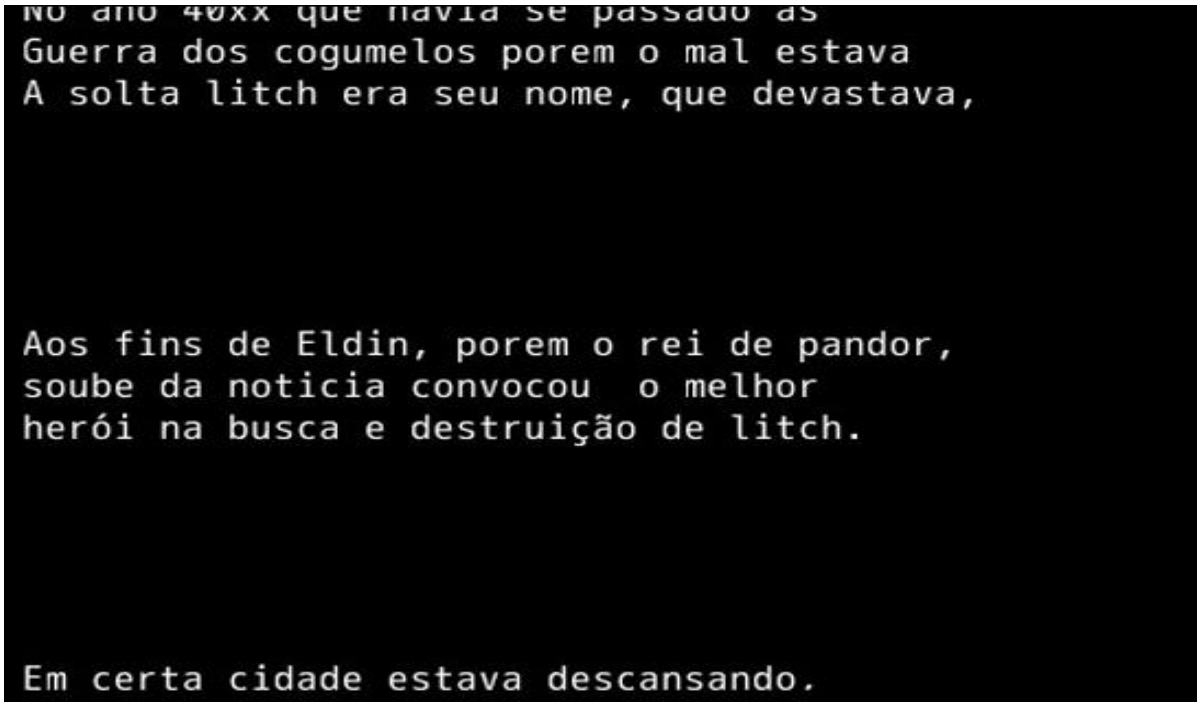

Figura 3 Tela da historia

7. DIAGRAMAS

7.1. DIAGRAMA DE SEQUENCIA

O tempo em cada tipo acontece no jogo.

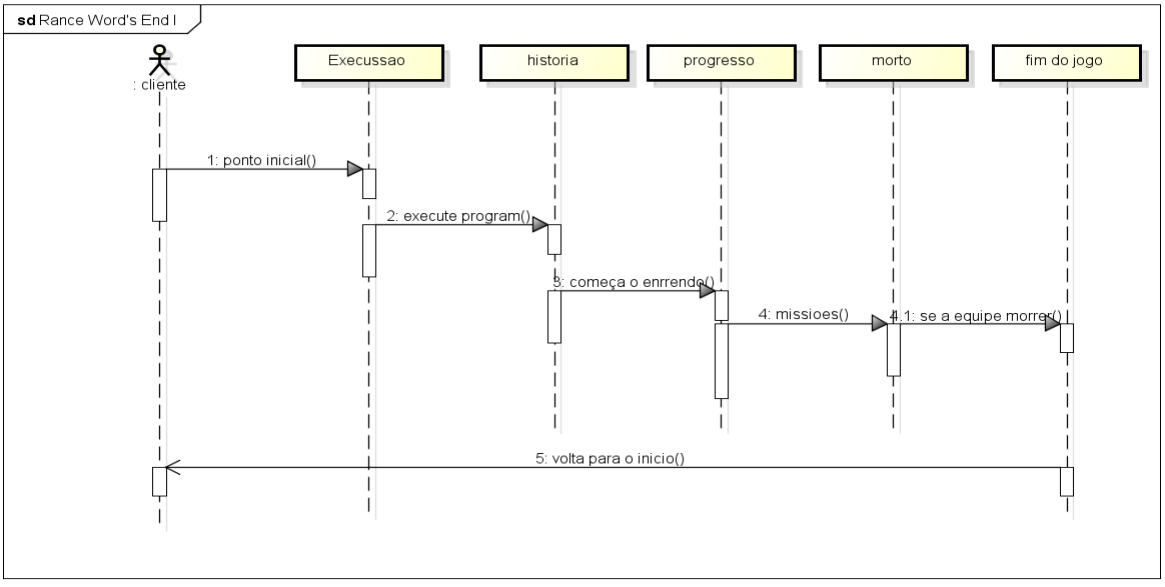

Figura 4 Diagrama de sequencia

8.2. DIGRAMA DE COMPONENTES

Aqui mostra no hardware e software de cada componente.

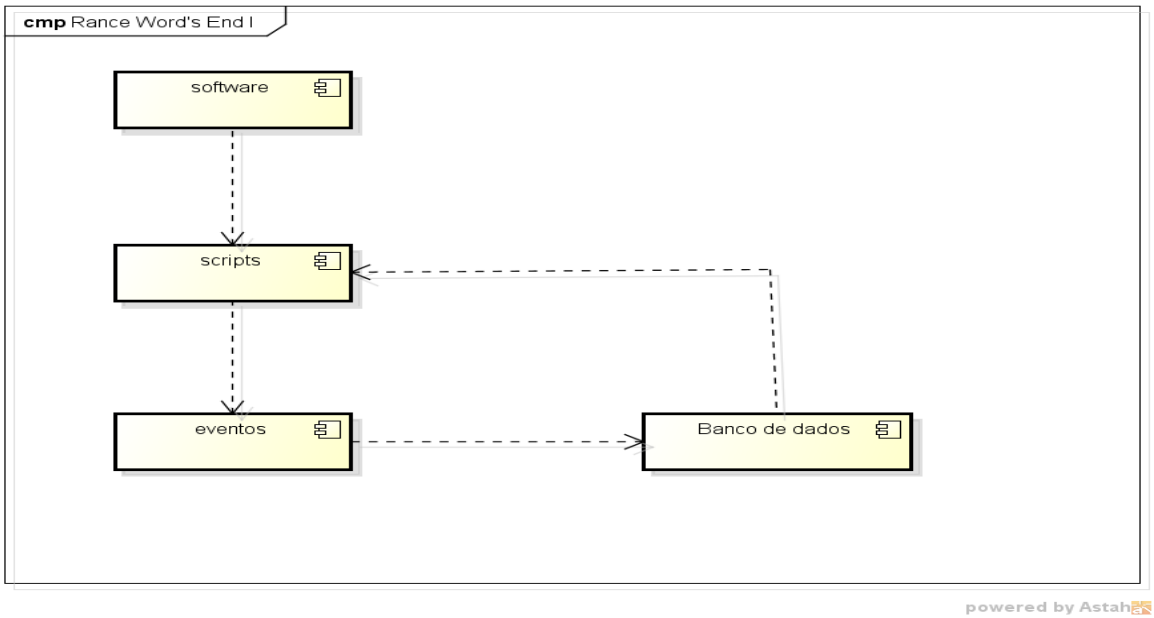

Figura 5 Diagrama de Componentes

8.3. DIAGRAMA DE IMPLEMENTAÇÃO

O diagrama se trata como dados via nuvem.

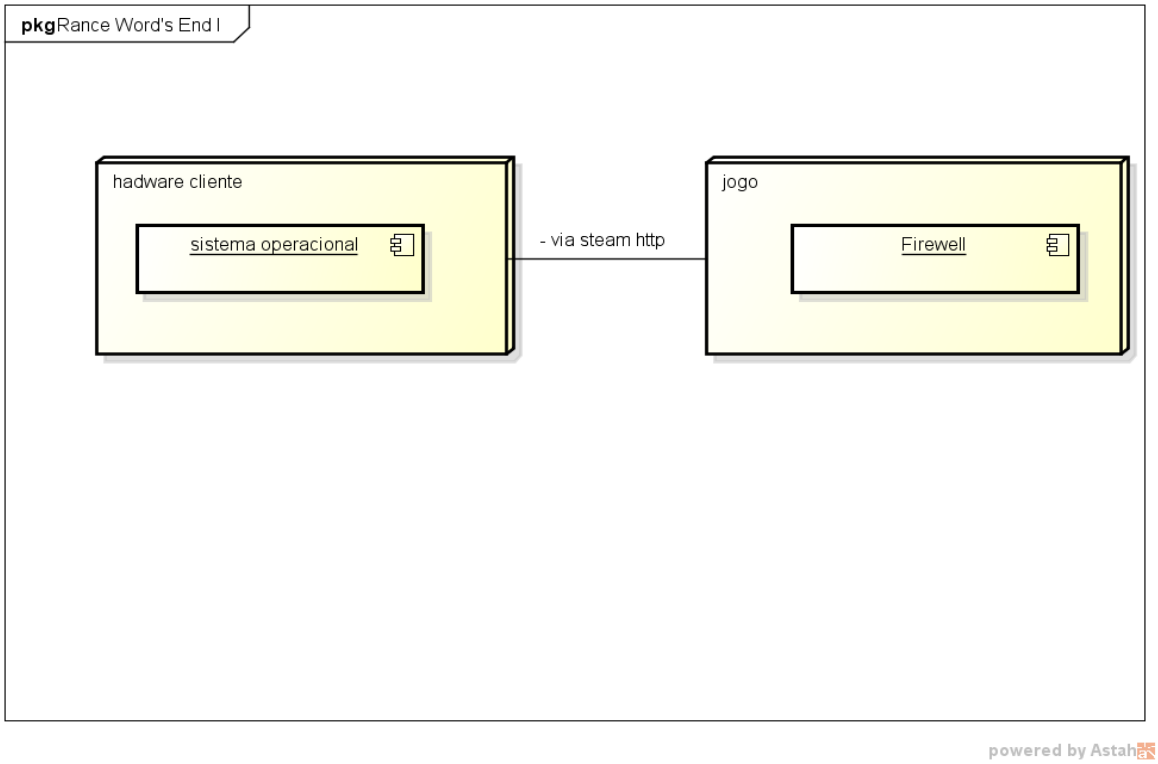

Figura 6 Diagrama Implementação

9. CONCLUSÃO

No desenvolvimento do estudo de uma análise de como o conteúdo dos jogos na educação que está sendo ministrado nas escolas, uma reflexão acerca dos benefícios didáticos pedagógicos e a dificuldade encontradas ao trabalhar neste conteúdo, até também permitiu utilizar diferentes métodos e avaliar como esses recursos auxiliam na aprendizagem do conteúdo. Em que a ferramenta será usada para incentivar usuário a aprender e a também ao desenvolvimento de R.P.G com relatos históricos da 1ª guerra mundial e logica da A.I (*Inteligência artificial*)

De modo geral, o interesse em jogos independentes, educativos em busca em usar eles como educacional, porém ainda possui algumas dificuldades em se fazer um game, ao estimular o interesse dos jogadores ou alunos para que aprendam de uma maneira diferente de se aprender, desde da guerra fria o começo do primeiro vídeo game criado em uma base militar até os dias atuais.

O jogo tem demonstrado muito ao conhecer ao jogador pelo tema e buscam se informar sobre o conteúdo, principalmente pela televisão e internet, diante, da história dos games ficou evidente que os objetivos de cada recurso didático foram realmente alcançados.

Os games didáticos forneceram aos sujeitos um ambiente enriquecedor e motivador em que além de divertir, passou a ser visto como um promotor de conhecer, permitindo que entendam melhor alguns conceitos, em que antes não foram processados, tirar dúvidas, revisar e reforçar o que foi visto no trabalho cientifico.

REFERÊNCIAS

CLUA, BITTENCOURT. DESENVOLVIMENTO DE JOGO 3D: CONCEPÇÃO, DESIGN E PROGRAMAÇÃO. 2005. 10 f. Dissertação (Mestrado) - Curso de Computação, Universidade de Computação, São Leopoldo, 2005. Cap. 3. Disponível em:<http://ddijogos.xpg.uol.com.br/desenvolvimento_de_jogos_3d_concepcao_design_e_programacao.pdf>. Acesso em: 28 mar. 2015.

CRUZ, **NÓVOA**, **ALBUQUERQUE**, **GAMES NA ESCOLA: CRIAÇÃO DE JOGOS ELETRONICOS COMO ETSRATEGIA DE LETRAMENTO DIGITAL.** 2012. 150 f. Tese (Doutorado) - Curso de Engenharia, Universidade Federal de Santa Catarina, Florianópolis, 2012. Disponível em: <stat.necat.incubadora.ufsc.br>. Acesso em: 28 mar. 2015.

FALCÃO; SANTOS; RODRIGUES. CORPO E MENTE: UMA ANALISE EDUCACIONAL E DE USABILIDADE DA INTERAÇÃO CORPORAL EM JOGOS DE VIDEOGAME. 2014. 10 f. Tese (Doutorado) - Curso de Estatística e Informática, Universidade Federal Rural de Pernambuco, Recife, 2014. Disponível em: <http://www.tacianapontual.com.br/wp-content/uploads/2015/01/PontualFalcao_et_al_IHC2014.pdf.>. Acesso em: 28 mar. 2015.

GEE. BONS VIDEO GAMES E BOA APRENDIZAGEM. 2009. 11 f. Tese (Doutorado) - Curso de Linguística, Stanford University, Arizona, 2009. Disponível em: <https://periodicos.ufsc.br/index.php/perspectiva/article/view/2175-795X.2009v27n1p167/14515>. Acesso em: 28 mar. 2015.

LUNA. COMUNICAÇAO E REPRESENTAÇÃO SOCIAL NOS VIDEOGAMES. 2008. 34 f. Tese (Doutorado) - Curso de Jornalismo, Universidade Federal da Paraíba, Paraíba, 2008. Disponível em: <http://www.insite.pro.br/2008/19.pdf>. Acesso em: 28 mar. 2015.

LUTERMAN. A INSCRIÇÃO DOS CORPOS EM UMA OUTRA DIMENSÃO: INTERATIVIDADE EM VÍDEO GAMES. 2012. 23 f. Tese (Doutorado) - Curso de Língua Portuguesa, Universidade Federal de Goiás, Inhumas, 2012. Disponível em: <http://www.revista.ueg.br/index.php/revelli/article/download/2902/1851>. Acesso em: 28 mar. 2015.

MAIKE, MIRANDA, BARANAUSKAS. INVESTIGANDO SOBRE REQUISITOS PARA UM JOGO DE RPG COM PROFESSORES DE UMA ESCOLA PÚBLICA DE ENSINO FUNDAMENTAL. 2011. 10 f. Tese (Doutorado) - Curso de Computação, Universidade Estadual de Campinas, São Paulo, 2011. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/viewFile/1618/1383>. Acesso em: 18 nov. 2016.

MEDEIROS. ADVERGAMES: A PUBLICIDADE EM JOGOS DIGITAIS COMO FORMA DE ATRAIR OCONSUMIDOR. 2010. 4 f. Tese (Doutorado) - Curso de Computação, Pontifícia Universidade Católica de Minas Gerais, Rio de Janeiro, 2010. Disponível em: <http://coral.ufsm.br/revistalappe/wp-content/uploads/2014/06/Advergames.pdf.>. Acesso em: 28 mar. 2015.

NASCIMENTO; ALCÂNTARA; CARVALHO. **ASPECTOS GERAIS DOS GAMES INDIES.** 2014. 12 f. Tese (Doutorado) - Curso de Comunicação em Mídias Digitais, Universidade Federal da Paraíba, Paraíba, 2014. Disponível em: <http://periodicos.ufpb.br/ojs2/index.php/tematica/article/view/20298>. Acesso em: 28 mar. 2015.

OLIVEIRA; PESSOA. **BENEFICIOS COGNITIVOS DOS VIDEOJOGOS: A PERCEPÇÃO DOS JOVENS ADULTOS.** 2008. 6 f. TCC (Graduação) - Curso de Psicologia, Universidade de Coimbra, Coimbra, 2008. Disponível em: <revistacomsoc.pt>. Acesso em: 28 mar. 2015.

ROSA, MALTEMPI. **RPG MAKER: UMA PROPOSTA PARA UNIR JOGO, INFORMÁTICA E EDUCAÇÃO MATEMÁTICA.** 2003. 20 f. TCC (Graduação) - Curso de Matemática, Universidade Estadual Paulista, Rio Claro, 2003. Disponível em: <http://www.rc.unesp.br/igce/demac/maltempi/Publicacao/Rosa-Maltempi-sipem03.pdf>. Acesso em: 18 nov. 2016.

SEIXAS; VERDOLINI; MARTINS. **OS BENEFICIOS DA INSERÇÃO DE GAMES ELETRONICOS NO PROCESSO DE ENSINO E APRENDIZAGEM NA LINGUA INGLESA.** 2011. 4 f. Tese (Doutorado) - Curso de Letras, Universidade Presbiteriana Mackenzie, São Paulo, 2011. Disponível em: <http://www.iiis.org/CDs2014/CD2014IMC/CICIC_2014/PapersPdf/CB656QG.pdf.>. Acesso em: 28 mar. 2015.

SEPÉ. **GAMES DE COMPUTADOR ON-LINE: REVISITANDO O CONSEITO DE COMUNIDADE NOS TEMPOS DA "AGORA ELETRONICA".** 2001. 9 f. Tese (Doutorado) - Curso de Língua Portuguesa, Universidade do Vale do Rio dos Sinos, Vale do Rio dos Sinos, 2001. Disponível em: <http://www.portcom.intercom.org.br/pdfs/63823526626657175643767741995>. Acesso em: 28 mar. 2015.

SILVA, FEITOSA, ROCHA. **A UTILIZAÇÃO DO RPG MAKER NA CONSTRUÇÃO DE RPGs EDUCACIONAIS.** 2009. 3 f. Dissertação (Mestrado) - Curso de Ciências da Computação, Universidade Federal Rural de Pernambuco, Pernambuco, 2009. Disponível em: <http://www.eventosufrpe.com.br/jepex2009/cd/resumos/R0614-1.pdf>. Acesso em: 18 nov. 2016.

SOHN, **HELOU**, **STEIL**. **NETWORK LEARNING: ESTUDO SOBRE CANAIS DE COMUNICAÇÃO QUE FALICITAM A APRENDIZAGEM EM REDE DE EMPRESAS DE GAMES.** 2012. 9 f. Tese (Doutorado) - Curso de Engenharia, Universidade Federal de Santa Catarina e Universidade do Vale do Itajaí, Florianópolis, 2012. Disponível em: <http://www.researchgate.net/profile/Andrea_Steil/publication/237082147_NETWORK_LEARNING_ESTUDO_SOBRE_CANAIS_DE_COMUNICAO_QUE_FALICITAM_A_APRENDIZAGEM_EM_REDE_DE_EMPRESAS_DE_GAMES/links/00b7d51b682a171c30000000.pdf>. Acesso em: 28 mar. 2015.

SOUSA. **UMA REVISÃO BIBLIGRAFICA SOBRE A UTILIZAÇÃO DO NINTENDO® WII COMO INSTRUMENTO TERAPEUTICO E SEUS FATORES DE RISCO.** 2011. 6 f. Dissertação (Mestrado) - Curso de Psicologia, Unesp - Campus de Bauru, Bauru, 2011. Disponível em: <periodicos.uem.br>. Acesso em: 28 mar. 2015.

SOUZA, KRAMER. O DEBATE PIAGET/VYGOTSKY E AS POLÍTICAS EDUCACIONAIS.1991. 12 f. Tese (Doutorado) – Curso de Pedagogia, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 1991. Disponível em: <http://www.fcc.org.br/pesquisa/publicacoes/cp/arquivos/965.pdf>>. Acesso em: 28 mar. 2015.

TAVARES. **INTELIGÊNCIA E VIDEOGAMES: O CORPO QUE LÊ.** 2012. 13 f. Tese (Doutorado) - Curso de Computação, Universidade Federal do Rio Grande do Norte – UFRN, Rio Grande do Norte, 2012. Disponível em: <http://www.researchgate.net/profile/Rogerio_Tavares2/publication/230651258_Inteligncia_e_Videogames_o_corpo_que_l/links/0912f50292a8a7c017000000.pdf>. Acesso em: 28 mar. 2015.

THIAGO, MENDES. **JOGOS DIGITAIS COMO OBJETOS DE APRENDIZAGEM: APONTAMENTOS PARA UMA METODOLOGIA DE DESENVOLVIMENTO.** 2011. 8 f. Tese (Doutorado) - Curso de Designer, Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, 2011. Disponível em: <http://www.sbgames.org/sbgames2011/proceedings/sbgames/papers/art/full/92067.pdf.>. Acesso em: 28 mar. 2015.

WEILLER. **JOGOS DIGITAIS: INTERFACES GRAFICAS E INTERAÇÃO.** 2010. 10 f. Tese (Doutorado) - Curso de Sociologia, Universidade de São Paulo, São Paulo, 2010. Disponível em: <http://www.insite.pro.br/2010/Outubro/jogos_digitais_interfaces.pdf>. Acesso em: 28 mar. 2015.

APENDICE A – CÓDIGO DE SELEÇÃO DE DIFICULDADE

```
#==============================================================
# JB{Seleção de Dificuldade} - Versão 1.2
#--------------------------------------------------------------
# Autor: JohnBolton
# Data : 23/06/2012
#--------------------------------------------------------------
# Descrição: Insere no jogo um sistema de dificuldade,
# onde a escolha de um certo nível resultará em uma
# alteração nos atributos dos inimigos, tornando-os mais
# fortes ou mais fracos. Uma das quatro opções só será
# liberada para o jogador se ele já tiver terminado o
# jogo pelo menos uma vez.
#--------------------------------------------------------------
# Instruções: Coloque-o acima do main e abaixo de todos
# os outros scripts, e por fim, personalize-o da maneira
# desejada no módulo de configurações abaixo.
#==============================================================

#==============================================================
# * Change_Log
#--------------------------------------------------------------
#
#  Versão 1.1
#
#    - Modificação em GOLD e EXP adicionados
#    - Correção nos tipos de dados
#
#  Versão 1.2
#
```

```
#    - Adicionado um cabeçalho
#    - Correção no número de arquivos zerados
#
#============================================================

module JohnBolton
 module Difficulty_System

   #============================================================#
   #    **              Configurações              **     #
   #============================================================#

# Opções que aparecerão na janela de seleção de dificuldade :
Options = [ 'Fácil', 'Normal', 'Difícil' ]

# Atributos dos monstros multiplicados respectivamente por :
JB_Some = [    1  ,   2    ,   3  ]

# Switch que dirá se o jogo já foi zerado ou não :
JB_SWITCH = 2

#Texto que aparecerá no cabeçalho :
Text = "            Selecione a dificuldade"

   #============================================================
   #    ----------     Fim das Configurações  -----------       #
   #============================================================

 end
end
```

```
#==============================================================
# ** Window_DifficultyCommand
#------------------------------------------------------------------------
#  Esta janela para seleção das opções Novo Jogo/Continuar na tela de título.
#==============================================================
class Window_DifficultyCommand < Window_Command
  include JohnBolton::Difficulty_System
  #--------------------------------------------------------------------------
  # * Inicialização do objeto
  #--------------------------------------------------------------------------
  def initialize
    @insane = false
    for i in 1..DataManager.savefile_max
      filename = i <= 9 ? "Save0#{i}.rvdata2" : "Save#{i}.rvdata2"
      filename = '#01010AE.bull' if @insane
      if File.file?(filename)
        file = File.open(filename, "rb")
        Marshal.load(file)
        extract_save_contents(Marshal.load(file))
        @insane = $game_switches[JB_SWITCH]
      end
    end
    super(0, 0)
    update_placement
    self.openness = 0
    self.active, self.visible = false, false
    open
  end
  #--------------------------------------------------------------------------
```

```
  # * Atualização da posição da janela
  #--------------------------------------------------------------------------
  def update_placement
    self.x = (Graphics.width - width) / 1.05
    self.y = (Graphics.height * 1.6 - height) / 2.8
  end
  #--------------------------------------------------------------------------
  # * Criação da lista de comandos
  #--------------------------------------------------------------------------
  def make_command_list
    add_command(Options[0], :easy)
    add_command(Options[1], :normal)
    add_command(Options[2], :hard)
  end
  #--------------------------------------------------------------------------
  # * Extrair conteúdo salvo
  #--------------------------------------------------------------------------
  def extract_save_contents(contents)
    $game_system        = contents[:system]
    $game_switches      = contents[:switches]
  end
end
#==============================================================
# ** Scene_Title
#------------------------------------------------------------------------------
#  Esta classe executa o processamento da tela de título.
#==============================================================
class Scene_Title < Scene_Base
  #--------------------------------------------------------------------------
  # * Inicialização do processo
```

```
#--------------------------------------------------------------------------
alias johnBolton_Difficulty_start start
def start
  johnBolton_Difficulty_start
  create_command_difficulty
  create_window_help
  close_difficulty_command
end
#--------------------------------------------------------------------------
# * Atualização da tela
#--------------------------------------------------------------------------
alias johnBolton_Difficulty_update update
def update
  johnBolton_Difficulty_update
  if Input.trigger?(:B) && !@difficulty_window.close?
    Sound.play_cancel
    close_difficulty_command
    @command_window.open
    @command_window.active, @command_window.visible = true, true
  end
end

def create_window_help
  @window_help = Window_Help.new(1)
  @window_help.set_text(JohnBolton::Difficulty_System::Text)
  @window_help.visible = false
  @window_help.opacity = 150
end
#--------------------------------------------------------------------------
# * Comando [Novo Jogo]
```

```
#--------------------------------------------------------------------------
def command_new_game
  open_difficulty_command
end
#--------------------------------------------------------------------------
# * Abre a janela de Dificuldade
#--------------------------------------------------------------------------
def open_difficulty_command
  @difficulty_window.visible, @difficulty_window.active = true, true
  @difficulty_window.open
  @window_help.visible = true
  @window_help.open
  close_command_window
end
#--------------------------------------------------------------------------
# * Fechamento da janela de Dificuldade
#--------------------------------------------------------------------------
def close_difficulty_command
  @difficulty_window.close
  @window_help.close
  update until @difficulty_window.close?
end
#--------------------------------------------------------------------------
# * Criação da janela de Dificuldade
#--------------------------------------------------------------------------
def create_command_difficulty
  @difficulty_window = Window_DifficultyCommand.new
  @difficulty_window.set_handler(:easy, method(:command_easy))
  @difficulty_window.set_handler(:normal, method(:command_normal))
  @difficulty_window.set_handler(:hard, method(:command_hard))
```

```
end
#--------------------------------------------------------------------------
# * Comando [Fácil]
#--------------------------------------------------------------------------
def command_easy
  new_game
  $game_system.vitamin_r(JohnBolton::Difficulty_System::JB_Some[0])
end
#--------------------------------------------------------------------------
# * Comando [Normal]
#--------------------------------------------------------------------------
def command_normal
  new_game
  $game_system.vitamin_r(JohnBolton::Difficulty_System::JB_Some[1])
end
#--------------------------------------------------------------------------
# * Comando [Díficil]
#--------------------------------------------------------------------------
def command_hard
  new_game
  $game_system.vitamin_r(JohnBolton::Difficulty_System::JB_Some[2])
end

#--------------------------------------------------------------------------
# * Inicia um novo jogo
#--------------------------------------------------------------------------
def new_game
  DataManager.setup_new_game
  close_command_window
  @difficulty_window.close
```

```
    fadeout_all
    $game_map.autoplay
    SceneManager.goto(Scene_Map)
  end
end

#==============================================================================
# ** Game_System
#------------------------------------------------------------------------------
#  Esta classe gerencia os dados relacionados ao sistema. Também gerencia
# veículos, BGM, etc.
# A instância desta classe é referenciada por $game_system.
#==============================================================================

class Game_System
  #--------------------------------------------------------------------------
  # * Variável pública
  #--------------------------------------------------------------------------
  attr_accessor :difficulty
  #--------------------------------------------------------------------------
  # * Definição do nível de dificuldade
  #--------------------------------------------------------------------------
  def vitamin_r(chevelle)
    @difficulty = chevelle
  end
end
#==============================================================================
# ** Game_Enemy
#------------------------------------------------------------------------------
```

```
#  Esta classe gerencia os inimigos. Ela é utilizada internamente pela
# classe Game_Troop ($game_troop).
#==============================================================

class Game_Enemy < Game_Battler
  #--------------------------------------------------------------------------
  # * [Alias] Aquisição do valor base do parâmetro
  #--------------------------------------------------------------------------
  alias jb_difsys_param_base param_base
  def param_base(*args)
    (jb_difsys_param_base(*args) * $game_system.difficulty).to_i
  end
  #--------------------------------------------------------------------------
  # * [Alias] Aquisição de experiência
  #--------------------------------------------------------------------------
  alias jb_difsys_exp exp
  def exp
    (jb_difsys_exp * $game_system.difficulty).to_i
  end
  #--------------------------------------------------------------------------
  # * [Alias] Aquisição de dinheiro
  #--------------------------------------------------------------------------
  alias jb_difsys_gold gold
  def gold
    (jb_difsys_gold * $game_system.difficulty).to_i
  end
end
```

APENDICE B – CÓDIGO SISTEMA BÁSICO DE MOUSE

```
#Basic Mouse System v2.7d
#----------#
#Features: Provides a series of functions to find the current x, y position of
#         the mouse and whether it is being clicked or not (left or right click)
#
#Usage:   Script calls:
#         Mouse.pos?    - returns the x, y position as an array
#         Mouse.lclick?(repeat) - returns if left click is achieved
#                                 repeat = true for repeated checks
#         Mouse.rclick?(repeat) - same as above for right click
#         Mouse.within?(rect) - passes a Rect through to check if cursor
#                               is within it, returns true if so
#
#        Events:
#         The following are placed in the name of an event:
#         &&  -  event can be triggered from afar by mouse click
#         I:# -  where # is the icon_index to change the cursor on hover
#
#        Example: I:262
#
#----------#
#-- Script by: V.M of D.T
#
#- Questions or comments can be:
#    given by email: sumptuaryspade@live.ca
#    provided on facebook: http://www.facebook.com/DaimoniousTailsGames
#   All my other scripts and projects can be found here:
#        http://daimonioustails.weebly.com/
```

```
#
#--- Free to use in any project, commercial or non-commercial, with credit given
# - - Though a donation's always a nice way to say thank you~ (I also accept actual thank you's)

#MOUSE_ICON, set to the index of the icon to use as a cursor
MOUSE_ICON = 147
CURSOR_OFFSET_X = 0
CURSOR_OFFSET_Y = 0

#Keeps cursor sprite within the game window
MOUSE_KEEP_WINDOW = true

#Whether clicking requires cursor to be within window or not
MOUSE_CLICK_WITHIN = false

#Whether to use 8 directional movement or not
MOUSE_DIR8 = false

#Use the Mouse Button Overlay:
USE_MOUSE_BUTTONS = false
#And here is where you set up your buttons! Simple overlay:
#(Picture files are to be stored in System)
#
# [ x , y, "filename", "script call when left clicked" ]
MOUSE_BUTTONS = [
        [0,416-32,"Shadow.png","SceneManager.call(Scene_Equip)"],
        [32,416-32,"Shadow.png","SceneManager.call(Scene_Item)"], ]

#Switch option to enable/disable the script
```

```
USE_MOUSE_SWITCH = false
MOUSE_SWITCH = 1

CPOS = Win32API.new 'user32', 'GetCursorPos', ['p'], 'v'
WINX = Win32API.new 'user32', 'FindWindowEx', ['l','l','p','p'], 'i'
ASKS = Win32API.new 'user32', 'GetAsyncKeyState', ['p'], 'i'
SMET = Win32API.new 'user32', 'GetSystemMetrics', ['i'], 'i'
WREC = Win32API.new 'user32', 'GetWindowRect', ['l','p'], 'v'

SHOWMOUS = Win32API.new 'user32', 'ShowCursor', 'i', 'i'
SHOWMOUS.call(0)

module Mouse
  def self.setup
    @enabled = true
    @delay = 0
    bwap = true if SMET.call(23) != 0
    bwap ? @lmb = 0x02 : @lmb = 0x01
    bwap ? @rmb = 0x01 : @rmb = 0x02
  end
  def self.update
    return false unless @enabled
    return false if USE_MOUSE_SWITCH && $game_switches[MOUSE_SWITCH]
    self.setup if @lmb.nil?
    @delay -= 1
    @window_loc = WINX.call(0,0,"RGSS PLAYER",0)
    if ASKS.call(@lmb) == 0 then @l_clicked = false end
    if ASKS.call(@rmb) == 0 then @r_clicked = false end
    rect = '0000000000000000'
    cursor_pos = '00000000'
```

```
  WREC.call(@window_loc, rect)
  side, top = rect.unpack("ll")
  CPOS.call(cursor_pos)
  @m_x, @m_y = cursor_pos.unpack("ll")
  w_x = side + SMET.call(5) + SMET.call(45)
  w_y = top + SMET.call(6) + SMET.call(46) + SMET.call(4)
  @m_x -= w_x; @m_y -= w_y
  if MOUSE_KEEP_WINDOW
    @m_x = [[@m_x, 0].max,Graphics.width-5].min
    @m_y = [[@m_y, 0].max,Graphics.height-5].min
  end
  return true
end
def self.pos?
  return[-50,-50] unless self.update
  return [@m_x, @m_y]
end
def self.lclick?(repeat = false)
  return unless self.update
  return false if @l_clicked
  if ASKS.call(@lmb) != 0 then
    @l_clicked = true if !repeat
    return true end
end
def self.rclick?(repeat = false)
  return unless self.update
  return false if @r_clicked
  if ASKS.call(@rmb) != 0 then
    @r_clicked = true if !repeat
    return true end
```

```
  end
  def self.slowpeat
    return unless self.update
    return false if @delay > 0
    @delay = 120
    return true
  end
  def self.within?(rect)
    return unless self.update
    return false if @m_x < rect.x or @m_y < rect.y
    bound_x = rect.x + rect.width; bound_y = rect.y + rect.height
    return true if @m_x < bound_x and @m_y < bound_y
    return false
  end
  def self.disable
    @enabled = false
    SHOWMOUS.call(1)
  end
  def self.enable
    @enabled = true
    SHOWMOUS.call(0)
  end
end

Mouse.setup

module DataManager
  class << self
    alias mouse_init init
  end
```

```
  def self.init
    mouse_init
    $cursor = Mouse_Cursor.new
  end
end

class Scene_Base
  alias cursor_update update_basic
  def update_basic
    cursor_update
    mouse_cursor
  end
  def mouse_cursor
    pos = Mouse.pos?
    $cursor.x = pos[0] + CURSOR_OFFSET_X
    $cursor.y = pos[1] + CURSOR_OFFSET_Y
  end
end

class Mouse_Cursor < Sprite_Base
  def initialize
    super
    @icon = MOUSE_ICON
    self.bitmap = Bitmap.new(24,24)
    draw_cursor
    self.z = 255
  end
  def set_icon(icon)
    return if @icon == icon
    @icon = icon
```

```
    draw_cursor
  end
  def draw_cursor
    self.bitmap.clear
    icon_bitmap = Cache.system("Iconset")
    rect = Rect.new(@icon % 16 * 24, @icon / 16 * 24, 24, 24)
    self.bitmap.blt(0, 0, icon_bitmap, rect)
  end
end

class Window_Selectable
  alias mouse_update update
  alias mouse_init initialize
  def initialize(x,y,w,h)
    mouse_init(x,y,w,h)
    @mouse_all_rects = []
    @timer = 0
  end
  def update
    mouse_update
    update_mouse if self.active
  end
  def update_mouse
    @timer -= 1
    @mouse_all_rects = []
    item_max.times {|i|
      rect = item_rect(i)
      rect.x += self.x + standard_padding - self.ox
      rect.y += self.y + standard_padding - self.oy
      if !self.viewport.nil?
```

```
        rect.x += self.viewport.rect.x - self.viewport.ox
        rect.y += self.viewport.rect.y - self.viewport.oy
      end
      @mouse_all_rects.push(rect) }
    item_max.times {|i|
      next if @timer > 0
      next unless Mouse.within?(@mouse_all_rects[i])
      @timer = 10 if i > top_row * 2 + page_item_max - 1
      @timer = 10 if i < top_row * 2
      self.index = i }
    process_cancel if Mouse.rclick? && cancel_enabled?
    return if MOUSE_CLICK_WITHIN && !within_index
    process_ok if Mouse.lclick? && ok_enabled?
  end
  def within_index
    item_max.times {|i|
      return true if Mouse.within?(@mouse_all_rects[i]) }
    return false
  end
end

class Window_NameInput
  alias mouse_process_handling process_handling
  def process_handling
    mouse_process_handling
    process_back if Mouse.rclick?
  end
  def item_max
    return 90
  end
```

```
end

class Window_Message < Window_Base
  def input_pause
    self.pause = true
    wait(10)
    Fiber.yield until Input.trigger?(:B) || Input.trigger?(:C) || Mouse.lclick? #if
!SceneManager.scene_is?(Scene_Map))
    Input.update
    self.pause = false
  end
end

class Scene_File < Scene_MenuBase
  alias mouse_update update
  def update
    mouse_update
    mouse_input
  end
  def mouse_input
    xx = 0
    yy = 56
    width = Graphics.width
    rectcm1 = Rect.new(xx, yy, width, savefile_height)
    rectcm2 = Rect.new(xx, yy + rectcm1.height, width, savefile_height)
    rectcm3 = Rect.new(xx, yy + rectcm1.height * 2, width, savefile_height)
    rectcm4 = Rect.new(xx, yy + rectcm1.height * 3, width, savefile_height)
    rectttl = Rect.new(xx, yy, width, rectcm1.height * 4)
    rectcmA = Rect.new(0, yy - 12, Graphics.width, 24)
    rectcmB = Rect.new(0, Graphics.height - 12, Graphics.width, 24)
```

```
    @scroll = self.top_index
    last_index = @index
    @index = (0 + @scroll) if Mouse.within?(rectcm1)
    @index = (1 + @scroll) if Mouse.within?(rectcm2)
    @index = (2 + @scroll) if Mouse.within?(rectcm3)
    @index = (3 + @scroll) if Mouse.within?(rectcm4)
    cursor_down(false) if Mouse.within?(rectcmB) and Mouse.slowpeat
    cursor_up(false) if Mouse.within?(rectcmA) and Mouse.slowpeat
    if @index != last_index
      Sound.play_cursor
      @savefile_windows[last_index].selected = false
      @savefile_windows[@index].selected = true
    end
    on_savefile_ok if Mouse.lclick? and Mouse.within?(rectttl)
    on_savefile_cancel if Mouse.rclick? and Mouse.within?(rectttl)
  end
end

class Scene_Gameover
  alias mouse_update update
  def update
    mouse_update
    goto_title if Mouse.lclick? or Mouse.rclick?
  end
end

class Game_Player < Game_Character
  alias mouse_move_update update
  def update
    mouse_move_update
```

```
  mouse_input
end
def mouse_input
  begin
  return if USE_MOUSE_BUTTONS && SceneManager.scene.mouse_overlay.update
  rescue
  return
  end
  return if !movable? || $game_map.interpreter.running?
  if !Mouse.lclick?(true) then return end
  if moving? then return end
  Graphics.width / 32 % 2 == 0 ? xxx = 16 : xxx = 0
  Graphics.height / 32 % 2 == 0 ? yyy = 16 : yyy = 0
  x = $game_map.display_x + (Mouse.pos?[0] + xxx) / 32
  y = $game_map.display_y + (Mouse.pos?[1] + yyy) / 32
  x -= 0.5 if Graphics.width / 32 % 2 == 0
  y -= 0.5 if Graphics.height / 32 % 2 == 0
  return if start_map_event_mouse(x, y, [0,1,2], false)
  if MOUSE_DIR8
    x = $game_map.display_x * 32 + Mouse.pos?[0]
    y = $game_map.display_y * 32 + Mouse.pos?[1]
    x -= @x * 32 + 16
    y -= @y * 32 + 16
    angle = Math.atan(x.abs/y.abs) * (180 / Math::PI)
    angle = (90 - angle) + 90 if x > 0 && y > 0
    angle += 180 if x < 0 && y > 0
    angle = 90 - angle + 180 + 90 if x < 0 && y < 0
    move_straight(8) if angle >= 337 || angle < 22
    move_diagonal(6,8) if angle >= 22 && angle < 67
    move_straight(6) if angle >= 67 && angle < 112
```

```
    move_diagonal(6,2) if angle >= 112 && angle < 157
    move_straight(2) if angle >= 157 && angle < 202
    move_diagonal(4,2) if angle >= 202 && angle < 247
    move_straight(4) if angle >= 247 && angle < 292
    move_diagonal(4,8) if angle >= 292 && angle < 337
  else
    x = $game_map.display_x + Mouse.pos?[0] / 32
    y = $game_map.display_y + Mouse.pos?[1] / 32
    sx = distance_x_from(x)
    sy = distance_y_from(y)
    if sx.abs > sy.abs
      move_straight(sx > 0 ? 4 : 6)
      move_straight(sy > 0 ? 8 : 2) if !@move_succeed && sy != 0
    elsif sy != 0
      move_straight(sy > 0 ? 8 : 2)
      move_straight(sx > 0 ? 4 : 6) if !@move_succeed && sx != 0
    end
  end
end
def start_map_event_mouse(x, y, triggers, normal)
  return false if $game_map.interpreter.running?
  $game_map.events_xy(x, y).each do |event|
    next unless event.trigger_from_afar
    if event.trigger_in?(triggers)
      event.start
      return true
    end
  end
  return false
end
```

```
end

class Game_Event
  def trigger_from_afar
    return @event.name.include?("&&")
  end
  def mouse_icon?
    @event.name =~ /I:(\d+)/ ? $1.to_i : false
  end
end

class Scene_Map
  attr_accessor   :mouse_overlay
  alias mouse_update update
  alias mouse_overlay_init initialize
  def initialize(*args)
    mouse_overlay_init(*args)
    @mouse_overlay = Mouse_Overlay.new if USE_MOUSE_BUTTONS
    @last_mouse_x = -1
    @last_mouse_y = -1
  end
  def update
    mouse_update
    mouse_input_events
    update_mouse_icon
  end
  def mouse_input_events
    xx = $game_player.screen_x
    yy = $game_player.screen_y
    xx -= 16;
```

```
  recttop = Rect.new(xx - 6, yy - 80, 44, 48)
  rectrit = Rect.new(xx + 32, yy - 36, 48, 44)
  rectbot = Rect.new(xx - 6, yy, 44, 48)
  rectleft = Rect.new(xx - 48, yy - 38, 48, 44)
  mouse_action(8) if Mouse.within?(recttop)
  mouse_action(6) if Mouse.within?(rectrit)
  mouse_action(2) if Mouse.within?(rectbot)
  mouse_action(4) if Mouse.within?(rectleft)
  call_menu if Mouse.rclick? and !$game_map.interpreter.running?
end
def update_mouse_icon
  Graphics.width / 32 % 2 == 0 ? xxx = 16 : xxx = 0
  Graphics.height / 32 % 2 == 0 ? yyy = 16 : yyy = 0
  x = $game_map.display_x + (Mouse.pos?[0] + xxx) / 32
  y = $game_map.display_y + (Mouse.pos?[1] + yyy) / 32
  x -= 0.5 if Graphics.width / 32 % 2 == 0
  y -= 0.5 if Graphics.height / 32 % 2 == 0
  return if x == @last_mouse_x && y == @last_mouse_y
  @last_mouse_x = x
  @last_mouse_y = y
  events = $game_map.events_xy(x,y)
  icon = MOUSE_ICON
  events.each do |event|
    icon = event.mouse_icon? if event.mouse_icon?
  end
  $cursor.set_icon(icon)
end
def mouse_action(d)
  return if !Mouse.rclick?(true) || $game_map.interpreter.running?
  $game_player.set_direction(d)
```

```
    $game_player.check_action_event
  end
end

class Window_NumberInput
  OFS = 12
  WLH = 24
  alias mouse_update update
  def update
    mouse_update
    mouse_input if SceneManager.scene_is?(Scene_Map) and self.active
  end
  def mouse_input
    hold_rect = []
    xx = self.x + OFS
    yy = self.y + OFS
    width = 20
    rectttl = Rect.new(xx, yy, self.contents.width, WLH)
    for i in Range.new(0, @digits_max - 1)
      hold_rect.push(Rect.new(xx, yy, width, WLH))
      xx += width
    end
    for i in Range.new(0, @digits_max - 1)
      @index = i if Mouse.within?(hold_rect[i])
    end
    rectok = Rect.new(xx, yy, 34, 24)
    rectnum = Rect.new(self.x + OFS, yy, @digits_max * 20, WLH)
    self.process_ok if Mouse.within?(rectok) and Mouse.lclick?
    process_mouse_change if Mouse.within?(rectnum)
  end
```

```
def refresh
  contents.clear
  change_color(normal_color)
  s = sprintf("%0*d", @digits_max, @number)
  @digits_max.times do |i|
    rect = item_rect(i)
    rect.x += 1
    draw_text(rect, s[i,1], 1)
  end
  draw_text(self.contents.width - 24, 0, 34, WLH, "OK")
end
def update_placement
  self.width = @digits_max * 20 + padding * 2 + 34
  self.height = fitting_height(1)
  self.x = (Graphics.width - width) / 2
  if @message_window.y >= Graphics.height / 2
    self.y = @message_window.y - height - 8
  else
    self.y = @message_window.y + @message_window.height + 8
  end
end
def process_mouse_change
  return unless active
  place = 10 ** (@digits_max - 1 - @index)
  n = @number / place % 10
  @number -= n * place
  if Mouse.lclick?
    n = (n + 1) % 10
    Sound.play_cursor
  end
```

```
    if Mouse.rclick?
      n = (n + 9) % 10
      Sound.play_cursor
    end
    @number += n * place
    refresh
  end
end


class Mouse_Overlay
  def initialize
    @mouse_buttons = []
    MOUSE_BUTTONS.size.times do |i|
      @mouse_buttons[i] = Mouse_Button.new
      @mouse_buttons[i].x = MOUSE_BUTTONS[i][0]
      @mouse_buttons[i].y = MOUSE_BUTTONS[i][1]
      @mouse_buttons[i].bitmap = Bitmap.new("Graphics/System/" +
MOUSE_BUTTONS[i][2])
      @mouse_buttons[i].on_lclick = MOUSE_BUTTONS[i][3]
    end
  end
  def update
    @mouse_buttons.size.times do |i| @mouse_buttons[i].update end
    if Mouse.lclick?(true)
      @mouse_buttons.size.times do |i|
        if Mouse.within?(@mouse_buttons[i].current_rect?)
          @mouse_buttons[i].on_lclick_eval
          return true
        end
      end
```

```
    end
    return false
  end
  def refresh
  end
end

class Mouse_Button < Sprite_Base
  attr_accessor   :on_lclick
  def current_rect?
    Rect.new(x,y,width,height)
  end
  def on_lclick_eval
    eval(on_lclick)
  end
end

class Window_Base
  def rect
    Rect.new(self.x,self.y,self.width,self.height)
  end
end

class Scene_Options < Scene_MenuBase
  alias mouse_update update
  def update
    mouse_update
    update_mouse
  end
  def update_mouse
```

```
    create_rects unless @rects
    @rects.size.times do |i|
      @index = i if Mouse.within?(@rects[i])
    end
    if Mouse.lclick?
      if audio_index(@index)
        x = Mouse.pos?[0]
        return if x < 48+4
        return if x > 48+4+400
        value = (x - 48).to_f / 400
        $game_options.preset_volume(:master, value) if @window_index[@index] ==
:master
        $game_options.preset_volume(:bgm, value) if @window_index[@index] == :bgm
        $game_options.preset_volume(:se, value) if @window_index[@index] == :se
        @window_masterbar.refresh($game_options.master_volume)
        @window_bgmbar.refresh($game_options.bgm_volume)
        @window_sebar.refresh($game_options.se_volume)
        Sound.play_cursor
        $game_map.autoplay
      end
    end
  end
  def create_rects
    @rects = []
    WINDOW_ORDER.each do |sym|
      @rects.push(@window_masterbar.rect) if sym == :master
      @rects.push(@window_bgmbar.rect) if sym == :bgm
      @rects.push(@window_sebar.rect) if sym == :se
      @rects.push(@window_resolution.rect) if sym == :resolution
      @rects.push(@window_message.rect) if sym == :message_se
```

```
      @rects.push(@window_switch.rect) if sym == :switch
    end
    @rects.push(@window_command.rect)
  end
end

class Window_RecipeConfirm < Window_Selectable
  alias mouse_rec_update update
  def update
    mouse_rec_update
    update_mouse if self.active
  end
  def update_mouse
    @timer -= 1
    @mouse_all_rects = []
    @mouse_all_rects.push(Rect.new(self.x,self.y,self.contents.width*0.85,self.height))
    @mouse_all_rects.push(Rect.new(self.x +
self.contents.width*0.85,self.y,self.contents.width*0.25,self.height))
    if Mouse.rclick?
      if Mouse.within?(@mouse_all_rects[1])
        change_amount(-1)
      else
        process_cancel if cancel_enabled?
      end
    elsif Mouse.lclick?
      process_ok if ok_enabled? && Mouse.within?(@mouse_all_rects[0])
      change_amount(1) if Mouse.within?(@mouse_all_rects[1])
    end
  end
  def within_index
```

```
    item_max.times {|i|
      return true if Mouse.within?(@mouse_all_rects[i]) }
    return false
  end
end
```