Os Prog{amadores}

CINXPLODE

Projeto referente a cadeira de Introdução a Programação

DOCUMENTAÇÃO DO PROJETO

Corpo da equipe

Gabriel Jesus - GMSJ

Hugo Silva - HSS

Lucas Job - LJBA

Paulo Andrade - PVOA

Paulo Silva - PVFLS

Samuel Souza - SBJS

Avaliador/monitor

Heitor Sammuel

Objetivo geral do projeto

Desenvolver um jogo na linguagem C, intranet e com inteface gráfica através da biblioteca allegro.

Prazo estimado para conclusão do Projeto

Terça-feira dia 05 de dezembro de 2017.

Declaração do Escopo do Projeto

Nome do Projeto	**CinXplode**	**Data: 02/12/2017**

Resumo do Projeto

O jogo CINXPLODE é um passa-tempo, desenvolvido na linguagem C com uma interface gráfica permitida devido a implementação da *allegro*, e roda apenas em *linux*. Este é intranet e pode ser jogado por 4 pessoas em perfis diferentes através da interação servidor-cliente, onde os jogadores acessam o servidor que permite a comunicação entre eles e jogam em tempo real, sem a necessidade de turnos.

Objetivo

Nas últimas semanas, nosso objetivo foi desenvolver um jogo com conexão servidor-cliente, que possuísse uma interface gráfica amigável, além de ser interessante e cativante para os jogadores.

Metas

Nossas metas foram estimuladas pelo empenho em entregar um projeto que fizesse o tempo empregado ser ótimo. Assim, buscamos melhorar nosso convívio em equipe, uma tarefa não muito trivial, além de dividir o objetivo central em diversas tarefas, que geraram confiança e estímulo para a conclusão do todo.
Como meta principal foi acordado desenvolver o jogo, divididos em 4 partes:

- **Backpack:**

✓ Estabelecer a conexão servidor-client;
✓ Teste de envio e recepção das mensagens.

- **Airfly:**

✓ Desenvolver matriz do jogo;
✓ Condicionar as movimentações dos personagens;
✓ Implementação da explosão dos blocos;
✓ Implementação da "morte" dos jogadores.

- **Pré-Landing:**

✓ Fazer a junção do jogo com a allegro;
✓ Implementação de novas Sprites;
✓ Implementação dos audios no jogo;

- **Landing:**

- ✓ Criação da apresentação;
- ✓ Apresentação do Projeto;
- ✓ Analisar dados.

Riscos Identificados

O fato de todos estarem produzindo a mesma parte do projeto ao mesmo tempo, foi notado que poderia acarretar na não finalização do jogo no prazo estimado, devido divergências e repetições de partes do jogo já então criadas.

Solução

Encontrar uma forma de comunicar os jogadores, tanto entre si quanto com o servidor, foi um dos maiores problemas no desenvolvimento do projeto, visto que essa é uma das partes cruciais para que o jogo funcione como planejado. Para solucionar, trabalhamos durante dias, buscando aperfeiçoar as estruturas dos dados que seriam compartilhados, e que, ainda assim, minimizasse o tempo e o fluxo de informações.

Restrições

Para o desenvolvimento do Jogo CINXPLODE no windows, se faz necessário algumas alterações no servidor, tais quais ainda não temos disponíveis para dar continuidade a extensão do projeto.Devido implementação do áudio, tivemos de alterar o make_file original, adicionando assim duas flags para não comprometer a execução do programa, pois este no decorrer da execução tem nos retornado "NULL" proveniente do fato do arquivo não ser encontrado.

Estrutura do Projeto

Para garantir um maior controle sobre o desenvolvimento do projeto, desenvolvemos todo o código em funções, modularizando-o, assim, eventuais dificuldades seriam corrigidas com maior precisão. Além da repartição anteriormente mencionada, foi imprescindível a criação de dois arquivos, separados, mas totalmente interligados. A estruturação está definida como:

- gameclient.c: Neste arquivo existem todas as funções relacionadas com o envio e recebimento de mensagens, interpretação e lógica do jogo. Vale salientar que este difere do arquivo Client.c encontrado na pasta lib.
- gameserver.c: Assim como no anterior, este arquivo contém funções relacionadas com o envio e recebimento de mensagens, além da responsabilidade de atualizar todos os jogadores(clientes) sobre todas as mudanças que venham a ocorrer na execução da partida.

De uma maneira mais próxima:

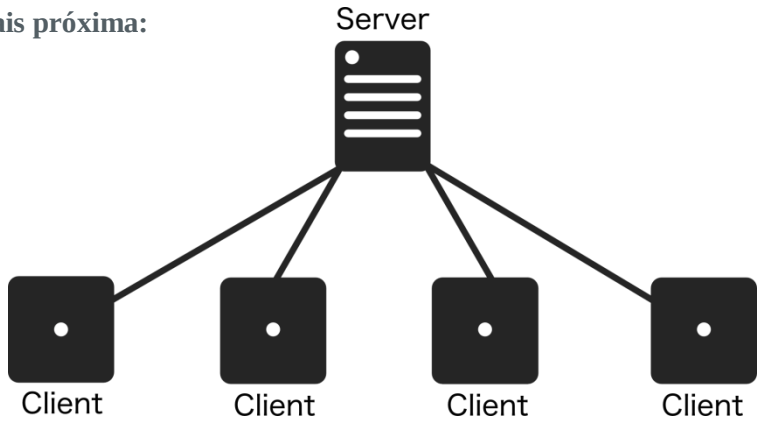


fig.1-Representação da comunicação entre servidor e cliente.

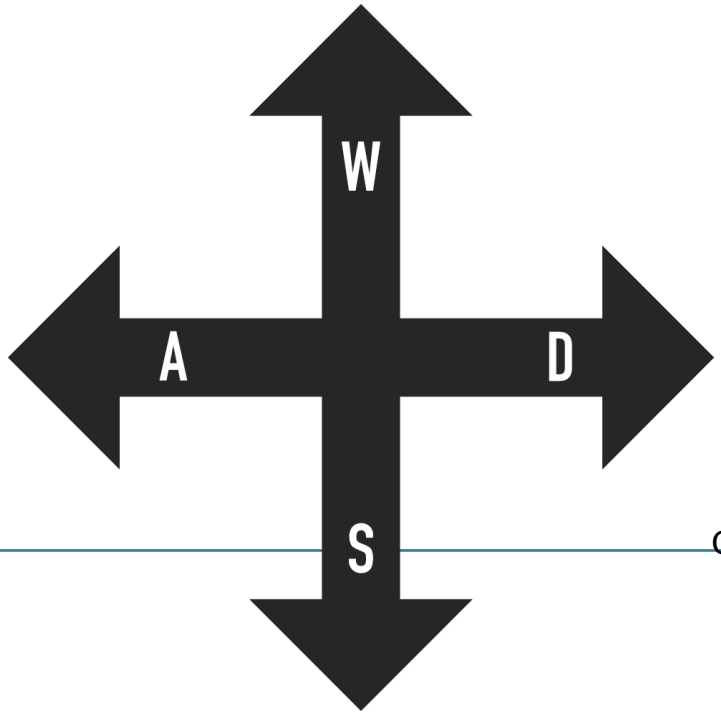
Antes de começar o desenvolvimento, precisávamos entender, mesmo que de uma forma superficial, o que estávamos fazendo. Para isso, estudamos e adquirimos uma noção sobre a arquitetura de comunicação que nosso projeto necessitava. Como demonstrado na **fig.1**, o envio de mensagens através do servidor pode ser propagado para todos os seus clientes(broadcast), ou para algum endereço especifico(através do identificador do destinatário - ID). Em posse desta ideia e da documentação das bibliotecas fornecidas para estabelecer a conexão, iniciamos o processo de estruturação do projeto.

Em seguida foi feito um esboço de como seria os mapas e os personagens, após isso foi criada as imagens que seriam utilizadas no gráfico do jogo, e após o estudo da allegro foi estabelecido limites na quantidade de pixels da imagem, onde tivemos de reduzir ao tamanho dos blocos da matriz.

Como cada participante da equipe fazia partes diferentes em tempos diferentes, sentimos a necessidade de utilizar uma ferramenta para a união das partes, dando inicio a utilização do github e em seguida do atom com o teletype para uma visualização em grupo do projeto.

Com a conexão servidor-client estabelecida, precisaríamos criar um limite para que o jogo pudesse prosseguir, este limite foi estabelecido como 4(quatro), para que assim o jogo só pudesse iniciar com o numero total de jogadores.

Cada dupla do grupo foi responsabilizada por entregar, dentro do prazo pré-determinado(o tempo dado a cada dupla sofreu variações devido à dificuldade que cada atividade apresentou), houve divisões na área de design, lógica do jogo e comunicação. Com a programação visual, surgiu a necessidade de desenhar cada elemento gráfico do jogo, desde a grama à sobreposição de pixels nos bonecos, utilizamos o Affinity Designer, GIMP e alguns softwares da Creative Cloud(Adobe) para otimizar o tempo, além de obter componentes apurados, com controle de cores e efeitos tridimensionais de relevo. No campo da lógica, nos deparamos com novas dificuldades. Por onde começar?! Como validar as informações?! Que estruturas usar para minimizar o fluxo de dados?! Para responder à esses questionamentos, recorremos aos nossos monitores, que nos mostraram que a forma mais eficaz seria tratar as tomadas de decisão em cada cliente, visto que a impossibilidade de movimentação evitaria o envio de uma mensagem para o servidor, guardando seu potencial para o que realmente seria necessário. Para verificar as intenções de movimento, usamos o retorno da função **getch( )**, direcionando-o para uma outra função, a **tratar_intencao( )**, nesta, olhamos para o mapa como uma matriz e realizamos a comparação da posição atual do jogador com o deslocamento em uma unidade, a depender dos botões pressionados, como ilustrado na **fig.2**.



Ainda na lógica, vale salientar que o controle do tempo de explosão de cada bomba é verificado, e, quando existe uma em estado iminente de detonação, o cliente busca nas estruturas(básica, do tipo jogador) por cada jogador no raio do evento, que corresponde a um deslocamento de uma unidade para todas as direções. Caso seja encontrado, um feedback é enviado para o servidor, que propaga para as demais conexões.

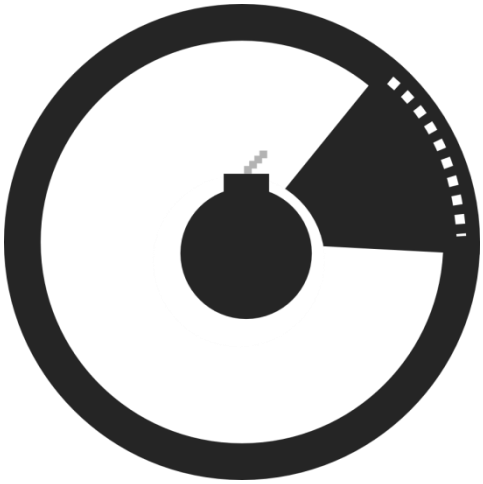


fig.3-Representação do raio de detonação.

Por fim, a equipe do servidor ficou responsável por ler, interpretar e implementar cada função da biblioteca fornecida, adequando seu código ao tipo de dados escolhido, bem como alterando o tamanho e tipo das variáveis contidas das estruturas. Aqui obtivemos uma menor taxa de aproveitamento em detrimento da dificuldade da escolha de que mensagens seriam transferidas. Agradecimento especial aos nossos monitores.