UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E SISTEMAS DIGITAIS

PCS3635 - LABORATÓRIO DIGITAL I

RELATÓRIO DA SEMANA 4 - POLI-ASTEROIDS

Felipe Luis Korbes - NUSP: 13682893
Henrique Eduardo dos Santos de Souza - NUSP: 13679972
João Felipe de Souza Melo - NUSP: 13682913

Turma: 5
Professor: Reginaldo Arakaki
Data da experiência: 03/04/2024

São Paulo
2024

Sumário

1. Introdução e Objetivos

O objetivo desta semana é de desenvolver as atividades do projeto, especialmente na implementação dos requisitos funcionais previstos pelo cronograma para essa última semana de projeto. Deve-se então partir para a finalização do projeto como um todo, com a lógica do jogo base completo concluída, envio e recebimento de dados pela conexão serial, e interface gráfico do jogo completa.

2. Progresso da Semana 4

Durante a semana 4, realizaram-se as últimas implementações do projeto, consolidando a lógica do jogo base e incorporando o tiro especial. Além disso, foram desenvolvidas duas unidades de controle adicionais para a transmissão de dados via conexão serial com o computador e outra para integrar a interface do computador com a placa FPGA.

Concomitantemente, foram elaborados os procedimentos de recebimento e processamento de dados pelo computador, visando à integração com a interface gráfica, a qual foi programada em Python utilizando o PyGame.

Paralelamente, confeccionou-se a maquete do jogo base, composta por uma caixa destinada à instalação da placa FPGA, bem como pelo controle físico do jogo, no qual os botões foram devidamente soldados em uma placa universal, os quais serão conectados através de fios na placa.

2.1 Parte Computacional em Verilog

Na última semana, a implementação mais recente relacionada ao jogo base foi a adição do tiro especial, que oferece ao jogador a capacidade de disparar em todas as direções simultaneamente, embora esteja disponível apenas ocasionalmente.

Além disso, houve uma adaptação na implementação dos níveis de dificuldade do jogo. Anteriormente, o jogador podia escolher entre diferentes níveis (fácil, médio e difícil), porém agora o jogo foi ajustado para ter apenas um modo de jogo. Neste novo modo, o jogo começa com uma dificuldade menor e vai aumentando gradativamente conforme o tempo passa. Essa mudança foi motivada pela percepção do grupo de que uma dificuldade estática poderia tornar o jogo

monótono, então optou-se por introduzir um aumento progressivo na velocidade para tornar a experiência mais dinâmica.

Adicionalmente, foi implementada outras duas unidades de controle, uma delas encarregada de transmitir os dados da placa FPGA para o computador por meio da serialização. Os dados enviados para o computador incluem: pontuação, número de vidas, posição da nave, informações sobre os asteroides, informações sobre os tiros e detalhes adicionais sobre o tiro especial, bem como o tipo de jogada realizada pelo jogador e um sinal indicando o fim do jogo.

A outra unidade de controle também é responsável por transmitir dados para a placa FPGA. Neste caso, estamos enviando os dados relativos ao menu principal do jogo, o qual foi implementado em Python e precisa receber informações sobre qual menu está sendo acessado a cada instante. Essa unidade de controle simplesmente indica em qual tela do menu inicial o jogador está navegando.

O diagrama atualizado das unidades de controle se encontra na Figura abaixo.

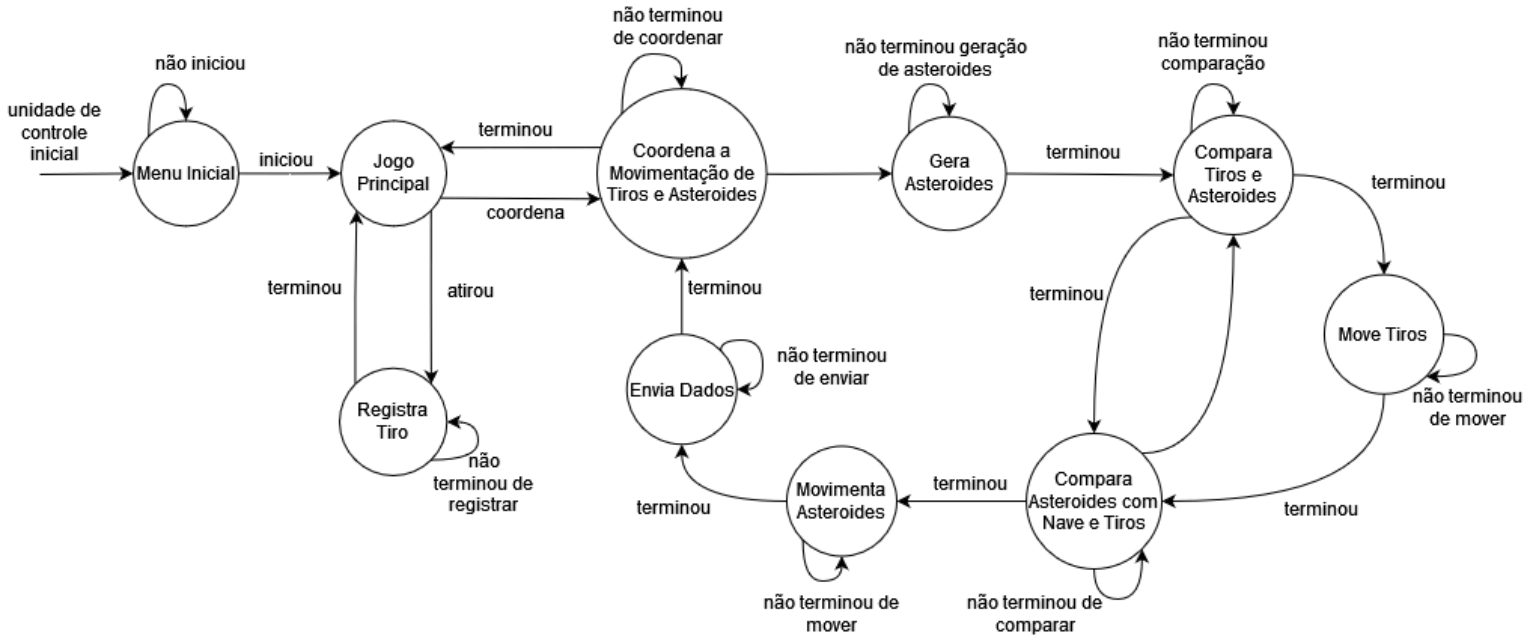

Figura 1 - Diagrama de transição das unidade de controle

2.2 Interface Gráfica no Pygame e Serialização

Com o sucesso da conexão serial, entre o computador e a placa FPGA, o grupo precisou montar uma infraestrutura para receber os dados do jogo, codificados na forma de uma sequência periódica de bytes, decodificá-los, interpretá-los e renderizá-los na tela.

Em primeiro lugar, o grupo definiu a codificação dos dados do jogo num bloco de dados, constituído por um total de 43 bytes de dados. Como o computador recebe continuamente os bytes do bloco, byte a byte, é necessária a inclusão de um indicador de fim de bloco, denominado de break point, que permite a seleção dos bytes corretos, correspondentes a um bloco de dados, sem erros de descompasso de bytes. Elegeu-se um break point de 2 bytes, logo a transmissão serial envia ao computador uma mensagem de 45 bytes, denominada no projeto como como chunk.

A cada byte, associou-se um Mnemônico para facilitar sua identificação no bloco. Por exemplo, o primeiro byte representa a pontuação do jogador é referenciado no projeto como PT. Os mnemônicos do bloco de dados são definidos como se segue:

PT: pontuação do player

G1: grupo 1, que incorpora:

direção da nave (bits[0:2])

vidas (bits[2:5])

dificuldade (bits[5:8])

AS: posições dos asteróides, sendo cada byte formado por:

x = bits[0:4]

y = bits[4:8]

DA: direções dos asteroides, sendo cada byte formado por direções de 4 asteroides:

direção do primeiro asteroide = bits[0:2]

direção do primeiro asteroide = bits[2:4]

direção do primeiro asteroide = bits[4:6]

direção do primeiro asteroide = bits[6:8]

TI: posições dos tiros, sendo cada byte formado por:

x = bits[0:4]

y = bits[4:8]

DT: direções dos asteroides, sendo cada byte formado por direções de 4 asteroides:

direção do primeiro tiro = bits[0:2]

direção do primeiro tiro = bits[2:4]

direção do primeiro tiro = bits[4:6]

direção do primeiro tiro = bits[6:8]

G2: grupo 2, que incorpora:

Jogada especial = bits[0]

Especial Disponível: bits[1]

Jogada Tiro: bits[2]

Tiro Disponível: bits[3]

parte vazia: bits[4:8]

BP: break point, escolhido como caracteres “AA” (ascii)= 0x41\x41 (hexadecimal).

	00	01	03	04	05	06	07	08
0000	PT	G1	AS[0]	AS[1]	AS[2]	AS[3]	AS[4]	AS[5]
0008	AS[6]	AS[7]	AS[8]	AS[9]	AS[10]	AS[11]	AS[12]	AS[13]
0010	AS[14]	AS[15]	DA[0]	DA[1]	DA[2]	DA[3]	TI[0]	TI[1]
0018	TI[2]	TI[3]	TI[4]	TI[5]	TI[6]	TI[7]	TI[8]	TI[9]
0020	TI[10]	TI[11]	TI[12]	TI[13]	TI[14]	TI[15]	DT[0]	DT[1]
0028	DT[2]	DT[3]	G2	BP[0]	BP[1]			

Total: 45 bytes

Figura 2 - Codificação do bloco de dados

O formato do bloco, assim como a sequência explícita dos bytes está representada na figura 2. Nela, a notação com colchete indica um byte específico de uma sequência de bytes, com o primeiro byte indexado como zero; por exemplo, AS[7] é a posição do oitavo asteroide.

Definida a codificação do bloco, projetou-se no software gráfico, feito em python, um buffer, que registra continuamente os bytes recebidos pela transmissão serial. O tamanho do buffer deve ser configurado para armazenar uma quantidade de bytes suficiente para captar certa quantidade de blocos de dados completos. Elegeu-se um bloco de 360 bytes como um tamanho suficiente para o buffer. Para detectar cada bloco, o buffer possui um detector de break points, que verifica se a transmissão serial enviou um breakpoint completo e íntegro. A cada breakpoint

recebido, o buffer realiza uma verificação para avaliar se o último chunk é um chunk válido, não corrompido.

Isso porque como os bytes do bloco de dados variam durante o jogo, muitos de forma aleatória, o breakpoint pode ocorrer naturalmente no bloco, numa certa posição. Por exemplo, uma certa combinação dos bytes PT e G1 poderia gerar um sequência de bytes interpretada como o break point elegido, no caso, AA (ascii) = 0x42 0x43 (hexadecimal). Por simplicidade, o grupo decidiu fazer uma verificação simplificada: o buffer apenas verifica se o break point atual e último break point detectado estão a uma distância coerente com o tamanho do chunk. Ou seja, para um bloco de 45 bytes, o último break point deve estar a 45 bytes de distância.

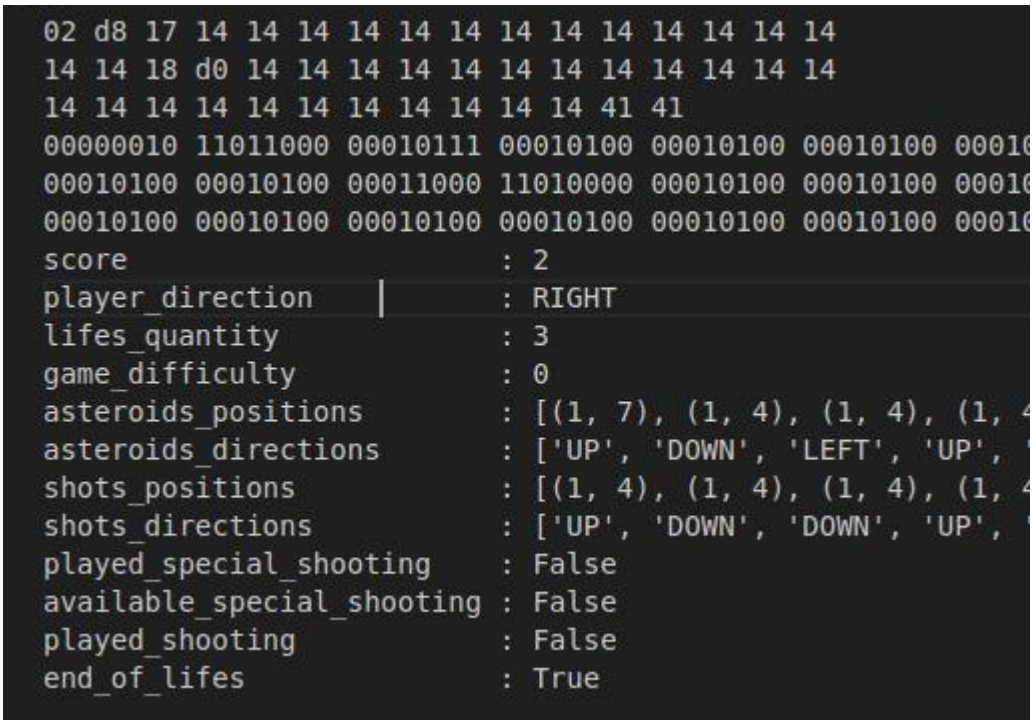

Figura 3 - bloco de dados decodificado no software

O software foi simulado para uma fita de bytes, que emula uma receção de bytes, tal qual a feita pela comunicação serial. O programa foi bem sucedido em registrar no buffer os bytes, em detectar os break points, em separar os chunks e em decodificar o bloco de dados em suas informações, armazenadas a partir daí em variáveis no python. O resultado da decodificação de um bloco está na Figura 3; as 6 primeiras linhas do arquivo de saída log.txt representam o buffer de 45 bytes, de forma

hexadecimal e em binário; as demais, os dados do bloco, já em seus tipos (inteiro, booleano, etc.).

Por fim, trabalhou-se na interface gráfica, que apenas interpreta o bloco de dados recebido pela transmissão serial. O grupo optou por reproduzir a estética simplificada do jogo original, com traços simples e retos, todos brancos contra fundo preto. Sob essas especificações, o grupo produziu uma interface que renderiza o bloco de dados. O resultado preliminar, para um bloco de dados aleatório, está contido na figura 4.

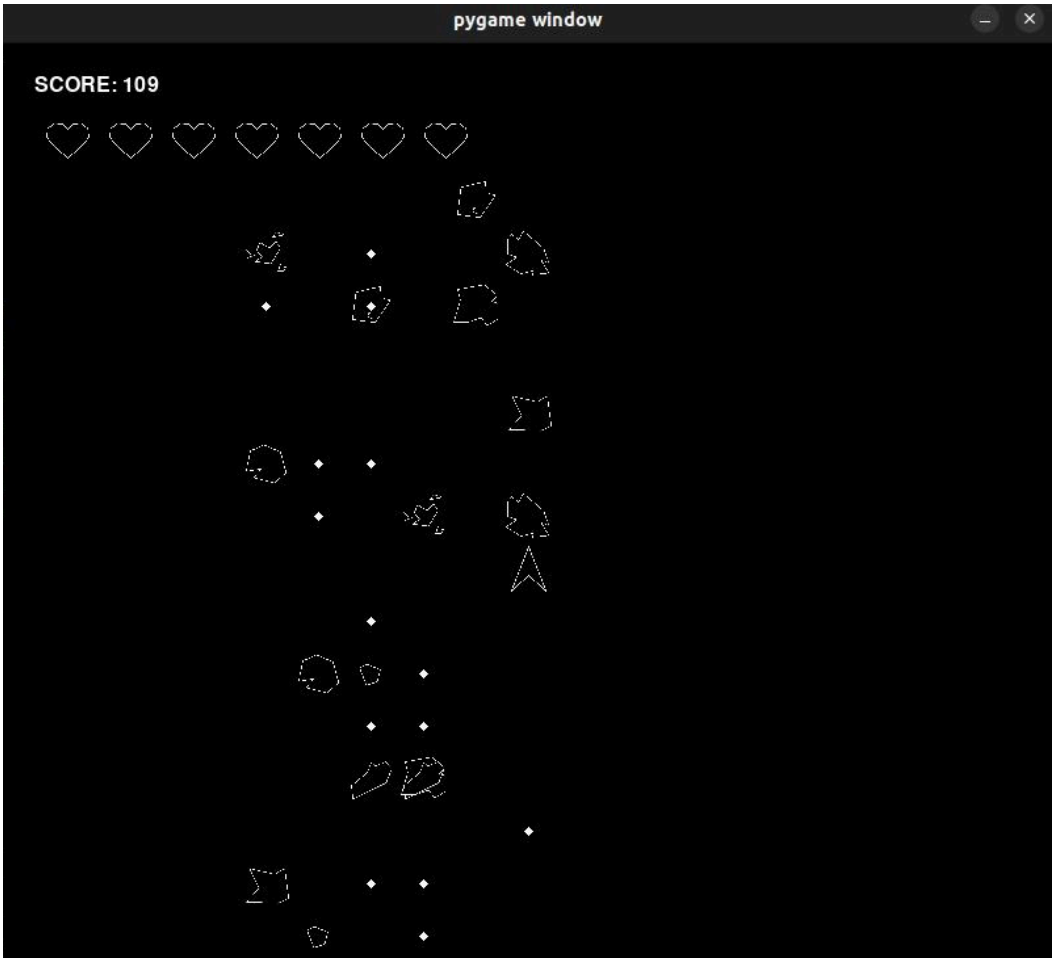

Figura 4 - Interface gráfica de gameplay (preliminar)

As imagens empregadas foram produzidas pelo próprio grupo. O grupo implementou uma interface com design responsivo, que emprega medidas de tamanho em unidades relativas. O sistema empregado utiliza uma escala de 0 a 100, em que 100 representa o tamanho total da tela. Assim, ao especificar um tamanho de um asteroide como 7x7, ele terá altura e largura iguais a 7% do

tamanho da tela, na horizontal e na vertical. Isso deve garantir que todas as telas renderizam um mesmo jogo, a menos de problemas com resolução.

2.3 Maquete e Controle

Na última semana, finalizou-se a maquete do projeto. Optou-se por utilizar uma caixa de papelão em vez de uma chapa de MDF, devido ao custo mais baixo e à redução do trabalho necessário. A chapa de MDF exigiria encomenda, modelagem 3D e cortes específicos, o que não seria viável dentro do prazo disponível. Assim, decoramos a caixa da melhor maneira possível, garantindo que atenda aos requisitos do projeto.

Além disso, construímos um controle para o jogo, foi soldado 6 botões em uma placa universal, seguindo um esquema pull-down. Os botões estão conectados por fios que serão ligados à placa FPGA.

2.4 Testes

Nesta última semana de projeto, o foco dos testes será na integração completa do projeto. Como a lógica do jogo já está operante, o objetivo principal é realizar testes finais para garantir que a serialização dos dados esteja funcionando corretamente. Para isso, será verificado a integridade dos bytes enviados para o computador. Além disso, será testado o funcionamento do controle que foi construído, assegurando que os botões estejam soldados corretamente na placa e que todas as conexões estejam adequadas.

Antecipa-se que a maior parte do tempo será dedicada a esses aspectos, dada a complexidade de testar a transmissão e recebimento de dados sem uma placa FPGA. Também será realizado testes finais na lógica do jogo, ajustando níveis de dificuldade e realizando melhorias finais para garantir que tudo esteja preparado para a apresentação na próxima semana.

3. Planejamento da aula prática

Inicia-se a discussão sobre o projeto e divisão das tarefas entre os membros do grupo, para que se demonstre os requisitos importantes do dia. Deve-se alinhar o cronograma, os requisitos prioritários em que o grupo vai trabalhar, tomar decisões de projeto, dentre outros aspectos importantes. Da parte de código, deve-se corrigir

eventuais erros que possam surgir durante a integração do circuito com o computador.

Espera-se que até o final da aula, esteja tudo integrado e que possa ser apresentado ao professor.

Para implementação na placa FPGA, definiu-se a associação de pinos e sinais entre a placa FPGA e a Analog Discovery tal qual na tabela abaixo.

Sinal	Pino na Placa DE0-CV	Pino na FPGA	Analog Discovery
CLOCK	GPIO_0_D0	PIN_M9	-
UP	GPIO_0_D1	PIN_T22	-
DOWN	GPIO_0_D2	PIN_N19	-
RIGHT	GPIO_0_D3	PIN_P19	-
LEFT	GPIO_0_D4	PIN_R22	-
ESPECIAL	GPIO_0_D5	PIN_M18	-
TIRO	GPIO_0_D6	PIN_P17	-
SAÍDA SERIAL	GPIO_1_D0	PIN_H16	-

Tabela 1 - Designação de Pinos

4. Relatório

Durante a aula prática, foram realizados os ajustes finais na maquete. Foram feitos furos na caixa para acomodar a placa, garantindo que ela fique oculta do jogador. Na parte externa da caixa, foram instalados os botões de controle que o jogador utilizará para interagir com o jogo, efetuando disparos e navegando pelo menu.

Figura 5 - Maquete do projeto

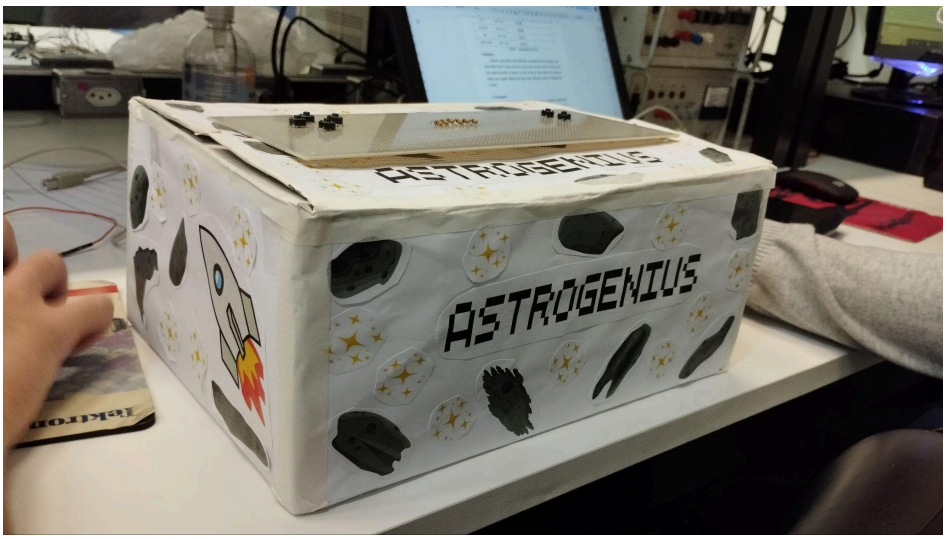

Figura 6 - Maquete do projeto

Ao mesmo tempo, foram feitas as últimas correções e ajustes de bugs na interface em Python. Essas alterações foram feitas para garantir que o programa seja capaz de receber e exibir os dados enviados pela placa para o computador por meio da transmissão serial.

5. Cronograma

Na última semana, conseguimos finalizar todos os requisitos estabelecidos para a entrega da semana 3. Também concluímos a maquete do jogo principal e o controle que os jogadores utilizarão. Agora, restam apenas pequenos ajustes finais e a integração completa do jogo na placa FPGA, juntamente com o computador e o

controle. Estamos focados em garantir que tudo funcione perfeitamente antes da apresentação final.

Cronograma do projeto Resumido	
Atividade	**Semana**
Finalizado - RF1 (Implementação Física na placa FPGA) Finalizado - RF2 (Controle da Nave Espacial) Finalizado - RF5 (Sistema de Geração de Asteroides) Finalizado - RF3 (Sistema de Vidas)	06/03 a 13/03
Finalizado - RF7 (Tiro Realizado) Finalizado - RF6 (Destruição do Asteroide) Finalizado - RF4 (Sistema de Pontuação) Finalizado - RNF3 (Escalabilidade) Finalizado - RNF4 (Manutenção) Finalizado - RF11 (Hitbox)	13/03 a 20/03
Finalizado - RF8 (Registro de Pontuação dos Players na memória) Finalizado - Implementação de 3 Níveis de Dificuldade Finalizado - RF9 (Menu de Interação) Finalizado - RF10 (Monitor)	20/03 a 27/03
Em andamento - Correções de bugs e implementações finais Em andamento - RFN5 (Som atrativo) Finalizado - Montagem do protótipo para o produto final	27/03 a 03/04