1. **Introdução:**
   1. **Orientação a Objetos**

A programação orientada a objetos(POO) é um dos principais paradigmas da programação. A orientação a objetos permite que os códigos sejam encapsulados em “pacotes” – que são conhecidos como classes –, o que torna possível a abstração e também uma melhor reutilização do código. Os objetos, responsáveis pelo nome “Orientada a Objetos”, são instâncias de uma classe, que possui métodos e variáveis específicos, com um valor “único” para cada objeto. Com a utilização do POO é possível criar tipos mais complexos de objetos, quebrando a barreira “imposta” pelos tipos primitivos, além de tornar seu código mais estruturado. (Anotações de aula)

* 1. **Polimorfismo e Herança**

Quando uma classe estende outra, ela herda todos os métodos e atributos que estão na classe mãe (ou superclasse), organizando assim as classes de forma em que a superclasse é a classe mais geral, e conforme existem suas “filhas”, as classes passam a ser cada vez mais específicas.

É possível sobrescrever os métodos da classe mãe, fazendo assim cad­­­a classe ter um comportamento diferente. Por exemplo, cães e gatos “estendem” mamíferos e fazem um som diferente, como se o método “fazSom()” tivesse sido sobrescrito em cada um deles.

O polimorfismo também pode ser exemplificado como dito acima. Ambas as classes (cães e gatos) herdaram o método fazSom() da superclasse(mamíferos) e, o fato desse método ter um resultado diferente em cada um deles (um cachorro latir e o gato miar) “ocorre” por conta do polimorfismo. (Anotações de aula)

* 1. **Multiprogramação**

Embora pareça, o computador ainda não é capaz de fazer duas, ou mais, coisas “ao mesmo tempo”. O que ocorre é o chaveamento dos processamentos no processador de forma tão rápida que esse chaveamento é imperceptível.

Tal ato porém foi perceptível no desenvolvimento do jogo, uma vez que era necessário remover um Asteroide da lista, enquanto essa era percorrida para imprimir seus elementos. Ao tentar fazer isso o erro de Concurrent Modification aparecia. Para resolver tal problema foram criadas as seções críticas no código. Nessas seções o processador é impedido de chavear o processamento, ou seja, somente executa um processo por vez. (Anotações de aula)

1. **Objetivos**

Aprender a utilizar os conceitos que foram ensinados na sala de aula – POO, multiprogramação, herança e polimorfismo – de forma prática. Desenvolver o jogo Asteroids do atari.

1. **Problemas**

No desenvolvimento do jogo, foi possível perceber que além do “problema” da lógica, existe também o problema de que desenvolver um jogo não é somente saber programar ou alguma linguagem, mas também realizar algoritmos para: descrever a física do jogo, detectar colisões, e o que fazer em cada colisão, além da movimentação de cada elemento do jogo.

* 1. **Física**

Por se passar no universo não existe atrito, portanto se algum elemento é acelerado ele não para de se mover até que seja imposta um força de mesma intensidade no sentido oposto do movimento. Os elementos, quando ultrapassam a tela, aparecem no outro canto, dando a ideia de que o universo do jogo é infinito.

* 1. **Colisões**

Os algoritmos de colisões foram feitos em duas partes. Para os Asteroides e os Misseis, se a distância das coordenadas de seu centro forem menores que a soma dos raios houve colisão, conforme mostrado na imagem 1. No caso da Nave, não existe colisão com o míssil, apenas com os Asteroides. A colisão da Nave com o Asteroide foi dividida em duas partes. A primeira realiza uma verificação igual à do Asteroide com a Nave.