Team Members:
Eli Gitelman, Adam Barson, and Austin Batistoni

The Asteroid Game

<Abstract>GamePiece
#space: Space -location: Point2D -shape: Shape -velocity: double -direction: Point2D
+<constructor> GamePiece(space: Space) +<pure virtual>setShape(): void +<pure virtual>move(): void +<pure virtual>explode(): void +<virtual>getLocation(): Point2D +<virtual>getDirection(): Point2D +<virtual>getSpace(): Space +<virtual>detectCollision(): bool

GamePiece is an abstract superclass which all other components of the game will extend. All GamePieces will have a location, shape, velocity, and direction. The shape will be determined in the constructors of all subclasses, while the location, velocity, and direction will be calculated dynamically, during runtime. The GamePiece class also contains several pure virtual methods, which each subclass will have to override, and virtual methods, which the subclasses will all have access to.

Pure Virtual Methods:

setShape: determines the Shape of the GamePiece.
move: defines the behavior for moving of the GamePiece.
explode: determines what happens when a GamePiece collides with something.

VIrtual Methods:

getLocation: return the location of the GamePiece.
getDirection: return the direction of the GamePiece.
getSpace: return the Space that the GamePiece is contained within.
detectCollision: return true if this GamePiece touches a different GamePiece

Bullet -> GamePiece
+BASE_VELOCITY: <static> double +LIFETIME: <static> int
+<constructor> Bullet(space: Space) +<override>setShape(): void +<override>move(): void +<override>explode(): void

A Bullet is generated whenever the Ship shoots. It represents a projectile that, upon contact, will destroy an asteroid. It will remains on screen until either it touches an asteroid, or is on screen for LIFETIME seconds.

Overridden Methods:

setShape: the shape of a Bullet will be a line.
move: movement will be determined by the direction of the ship.
explode: the line will be removed from the screen when a collision occurs.

Ship -> GamePiece
+TERMINAL_VELOCITY: <static> double +FRICTION: <static> double -numLives: int -thrustVelocity: double -rotationalVelocity: double
+<constructor> Ship(space: Space) +regenerate: void +rotateRight(): void +rotateLeft(): void +break(): void +shoot(): void +<override>setShape(): void +<override>move(): void +<override>explode(): void

The Ship is the avatar of the player. The Ship can move around the screen, using the arrow keys to change direction, the up key to thrust, and the down key to break. The Ship can also shoot Bullets to destroy incoming asteroids.

Methods:

regenerate: after the Ship is destroyed, it will regenerate to an empty place in Space.
rotateRight: when the right arrow key is pressed, the Ship rotates right.
rotateLeft: when the right arrow key is pressed, the Ship rotates left.

break: when the down arrow key is pressed, the Ship will slow down.
shoot: when the spacebar is pressed, the Ship will shoot a Bullet.

Overridden Methods:

setShape: the shape of a Ship will be a triangle.
move: movement will be determined by the direction of the ship, and will be called when the up arrow key is pressed.
explode: the Ship will generate an explosion image when a collision occurs.

Asteroid -> GamePiece
+GENERATION_RATE: <static> double -radius: double
+<constructor> Asteroid(space: Space, radius: double) +<override>setShape(): void +<override>move(): void +<override>explode(): void

Asteroids are the main way players get points in the game, more asteroids killed means more points. Asteroids will spawn at a random coordinate and explode upon being shot with a bullet, or colliding with the player. If the player collides with an asteroid he/she will lose a life. Asteroids have a set speed and can wrap around the "map" and when generated are directed towards the center of the map.

Overridden Methods:

setShape: the shape of an Asteroid will be a circle.
move: an Asteroid will move towards the center of the screen.
explode: an Asteroid will generate an explosion image when a collision occurs.

Space
-ship: Ship -bullets: vector<Bullet> -asteroids: vector<Asteroid>
+<constructor> Space() +drawAllAsteroids(): void +moveAllAsteroids(): void +moveAllBullets(): void +nextLevel(): void +pause(): void +openMenu(): void +highScore():void

Space is the "map" of the game. It holds all the pieces, has a pause button, highscore, and menu options as well. All game pieces will be "generated" on/in space.

Methods include highScore, openMenu, pause, nextLevel, as well as generating all of the asteroids, the ship, and the bullets shot by the ship.

End Game Goals:

After we have a functioning game of Asteroid, there are many extra little "tidbits" we'd like to add, from being able to change the color of one's ship/laser, to being able to upgrade the laser. We are even hoping to be able to have asteroids that break down into smaller asteroids as well as asteroids that seek out the player instead of moving in a random direction. At a base level we want the ship to be able to rotate and move, however we'd really like to be able to present the game as if the ship is in a legitimate vacuum. For instance, when a player would thrust, it'd accelerate the ship, which would continue to move in the given direction with very limited friction until the player decides to thrust in a different direction creating the feeling of flying in space.