# Basketball Random

Period 7
Nathaniel Chae, Orlin Dyankov, Turbold Gereltod
The Basketting Basketballers

## Description

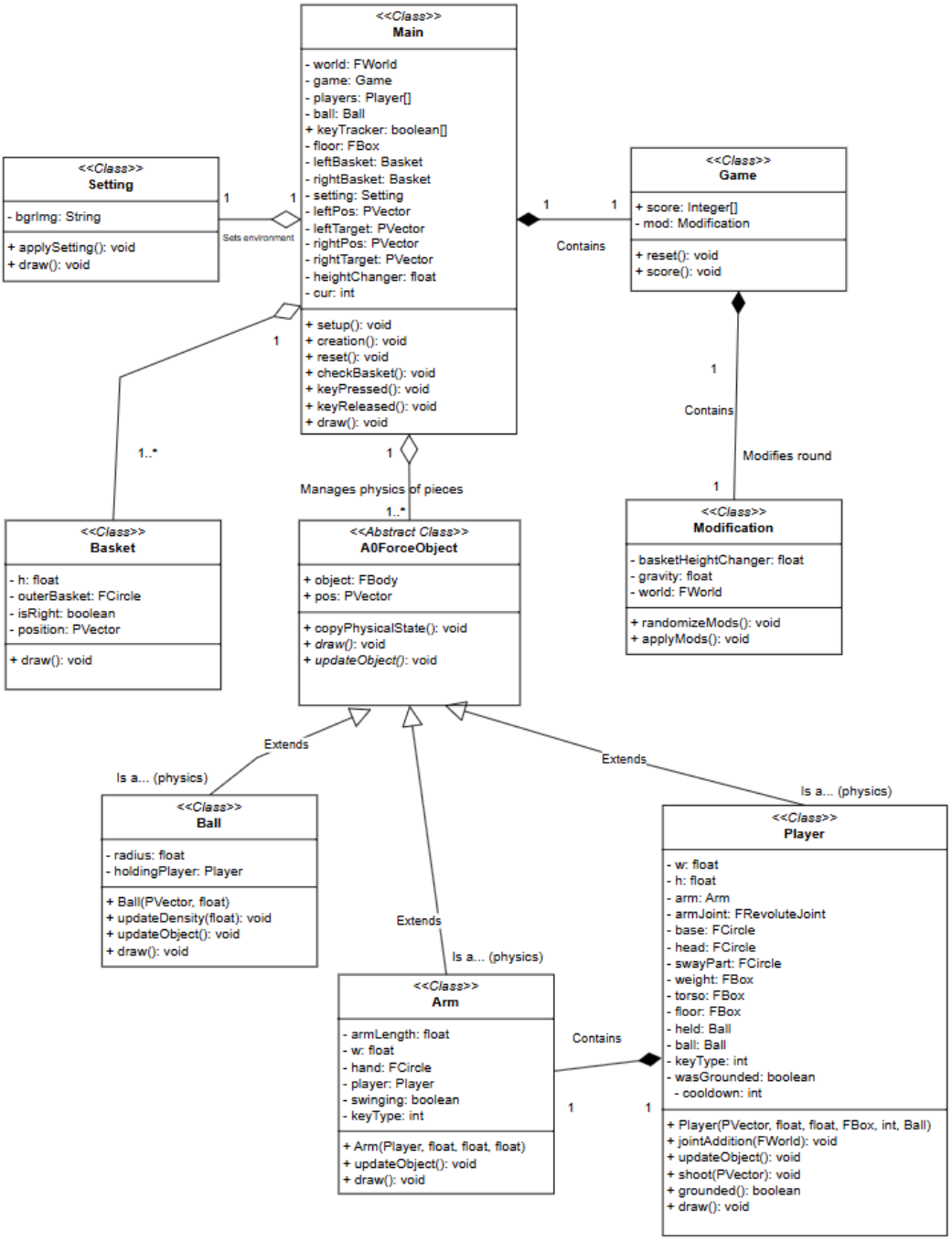
*A 2D 2-player chaotic basketball game with one input for each player: W and UP ARROW. These inputs control all of the player's movements, jumping, and swinging arms, which both contribute to shooting.*
*All players sway back and forth, allowing movement by timing jumps using the momentum of the sway.*

- User Input
    - W and Up Arrow command jumping, shooting, and arm swinging for **2 in-game** players simultaneously.
- Randomized Rounds
    - Modifications are randomly generated and applied each round.
- Randomized Physics
    - PVectors used to control components of force as well as variables used to modify fisica objects are randomized.
- Ball stealing, shooting, jumping
    - Collisions and physics are used to detect stealing and scoring.

**Requisite Libraries:**

- *Fisica* | *A wrapper for JBox2D, a 2D physics engine* - by Ricard Marxer
    - Handles all 2D physics simulations, including interactions between game elements.

# UML Diagram


<<Class>>
Main
- world: FWorld
- game: Game
- players: Player[]
- ball: Ball
+ keyTracker: boolean[]
- floor: FBox
- leftBasket: Basket
- rightBasket: Basket
- setting: Setting
- leftPos: PVector
- leftTarget: PVector
- rightPos: PVector
- rightTarget: PVector
- heightChanger: float
- cur: int
+ setup(): void
+ creation(): void
+ reset(): void
+ checkBasket(): void
+ keyPressed(): void
+ keyReleased(): void
+ draw(): void
<<Class>>
Setting
- bgrImg: String
+ applySetting(): void
+ draw(): void
Sets environment
<<Class>>
Game
+ score: Integer[]
- mod: Modification
+ reset(): void
+ score(): void
Contains
<<Class>>
Basket
- h: float
- outerBasket: FCircle
- isRight: boolean
- position: PVector
+ draw(): void
<<Abstract Class>>
A0ForceObject
+ object: FBody
+ pos: PVector
+ copyPhysicalState(): void
+ draw(): void
+ updateObject(): void
Manages physics of pieces
<<Class>>
Modification
- basketHeightChanger: float
- gravity: float
- world: FWorld
+ randomizeMods(): void
+ applyMods(): void
Modifies round
Contains
Extends
Is a... (physics)
<<Class>>
Ball
- radius: float
- holdingPlayer: Player
+ Ball(PVector, float)
+ updateDensity(float): void
+ updateObject(): void
+ draw(): void
<<Class>>
Arm
- armLength: float
- w: float
- hand: FCircle
- player: Player
- swinging: boolean
- keyType: int
+ Arm(Player, float, float, float)
+ updateObject(): void
+ draw(): void
<<Class>>
Player
- w: float
- h: float
- arm: Arm
- armJoint: FRevoluteJoint
- base: FCircle
- head: FCircle
- swayPart: FCircle
- weight: FBox
- torso: FBox
- floor: FBox
- held: Ball
- ball: Ball
- keyType: int
- wasGrounded: boolean
- cooldown: int
+ Player(PVector, float, float, FBox, int, Ball)
+ jointAddition(FWorld): void
+ updateObject(): void
+ shoot(PVector): void
+ grounded(): boolean
+ draw(): void

## How does it work?

**Main Objective:** Score 5 or more points before the other player.

The game will begin with four **in-game** players standing upright in front of their baskets. **Two players** will be under each basket, controlled by **one real player's** input.

The left-hand player uses the **W key**, the right-hand player uses the **up arrow**.

Input Response:
When the player holds down their input, both players on their team move their arms backward relative to where their head is facing.
If **they have the ball, letting go of the input will throw the ball.**
**A player cannot have possession twice in the span of 0.5 seconds.**
If the player is **on the ground** while pressing their input, their player will jump regardless of possession.

If the ball is in possession and the **opposing team's hand touches the ball**, they gain possession of the ball.

## Functionalities / Issues

Current functionalities:

- Implemented 2D physics for circle and rectangle
    - The circle will serve as a hitbox for the basketball, the rectangle will serve as hitboxes for the player, arm, and rim of the basketball hoop.
    - Associated Issue & Fix: Attempting to create rigid-body physics involving a realistic response to collisions, which would cause the body/bodies to rotate, was unachievable with a conventional geometric collision library. The solution was to use a physics library (fisica).
- Implemented a pill-shaped player hitbox which uses a disjoint weight to balance upright.
    - Associated Issue & Fix: The weight had to be disjoint and dense to balance; the physics engine does not handle density with a sufficient level of accuracy, and therefore the weight was moved outside of the base.
- Implemented an arm on the player which rotates upwards when a key is pressed.
    - Associated Issue & Fix: The arm had an ambient angular velocity which caused the player to tilt left by about 80 degrees when left to stand. A conditional which reset the angular velocity when the arm was at 0 degrees was implemented to fix it.
- Implemented Modifications class

    - Holds attributes to apply to game elements.
- Implemented Setting class to set background image.
- Implemented Game class to manage scoring
- Implemented Main file to handle resetting and detecting scoring, as well as initializing fisica objects.
- Issue and Fix: The head on the player has a weight when it is added to the player, this cannot be mitigated by decreasing its density to a low number. When shrinking the player, the weight of the head becomes negligent relative to the other parts of the player when balancing.
- Implemented scoring

## Log

- Turo
    - Remade the UML Diagram on a different platform with all feedback
    - Implemented Modifications class
    - Reworked UML to overhaul changes previously in the UML Diagram
    - Reworked Modification so that it would apply mods to other classes directly from the Modification class
    - Made 1st iteration of basket
    - Restructured Main and Game
    - Made modifications base for implementation with Main
    - Vetted final UML
- Nathaniel
    - Implemented Player, Ball, and associated physics
    - Implemented hand / pick up, throw of arm and hand.
    - Wrote the design document
    - Implemented Main file
    - Implemented Basket
- Orlin
    - Implemented Game and Setting class
    - Implemented swinging arm
    - Started to implement outfits
    - Reworked Arm
    - Implemented swaying of player upon landing
    - Vetted all classes and code