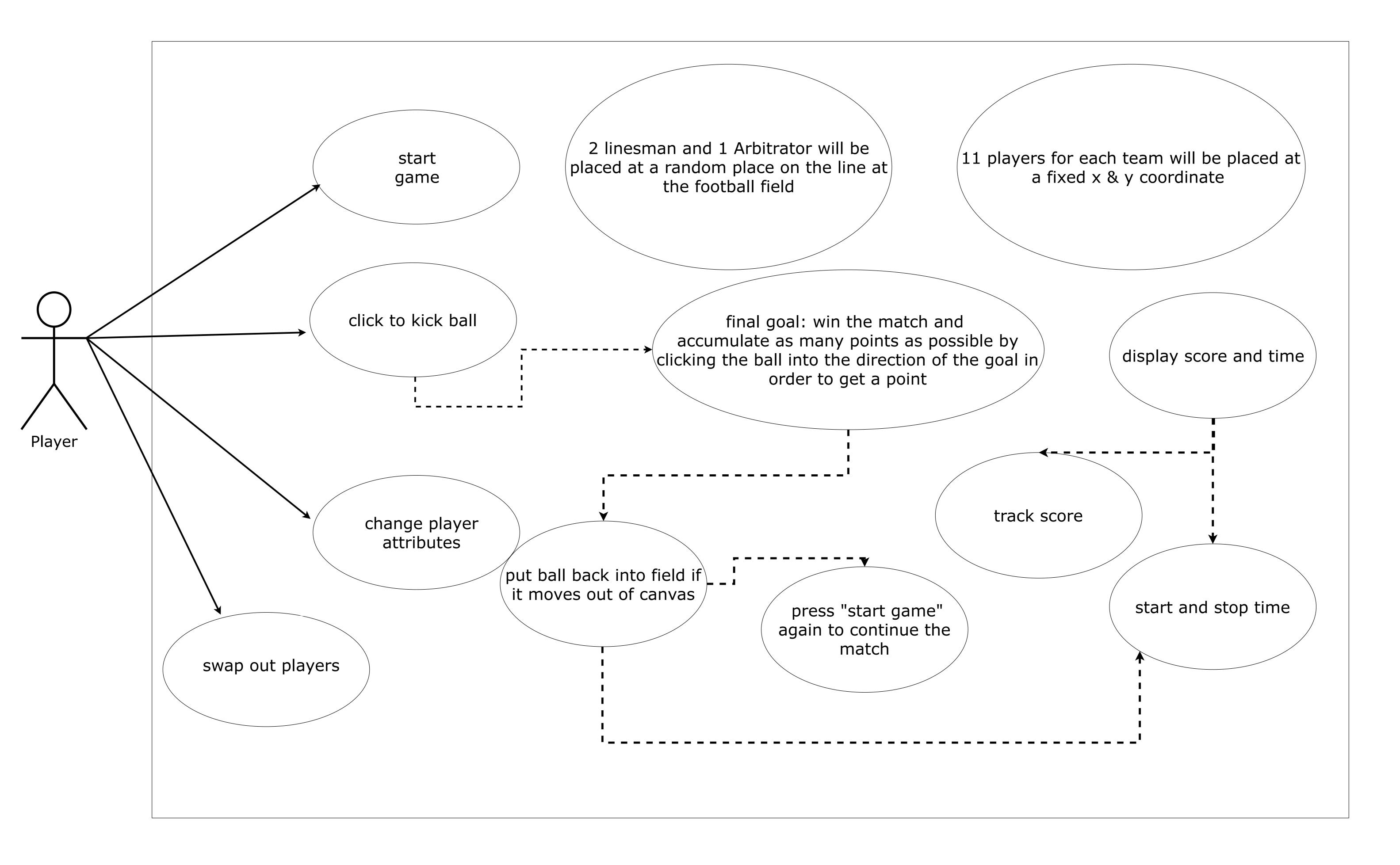
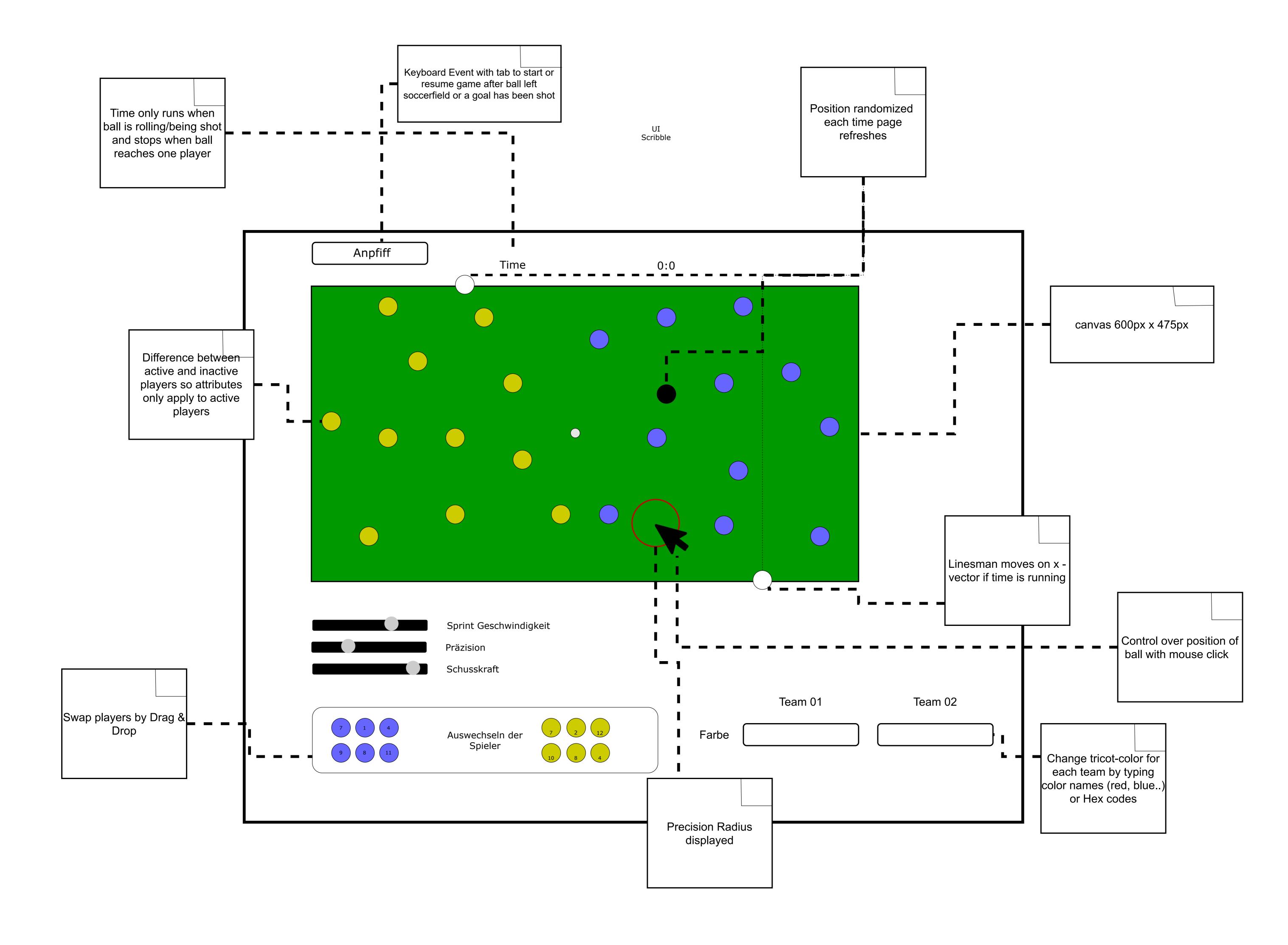
Use-Case-Diagram





Klassendiagram

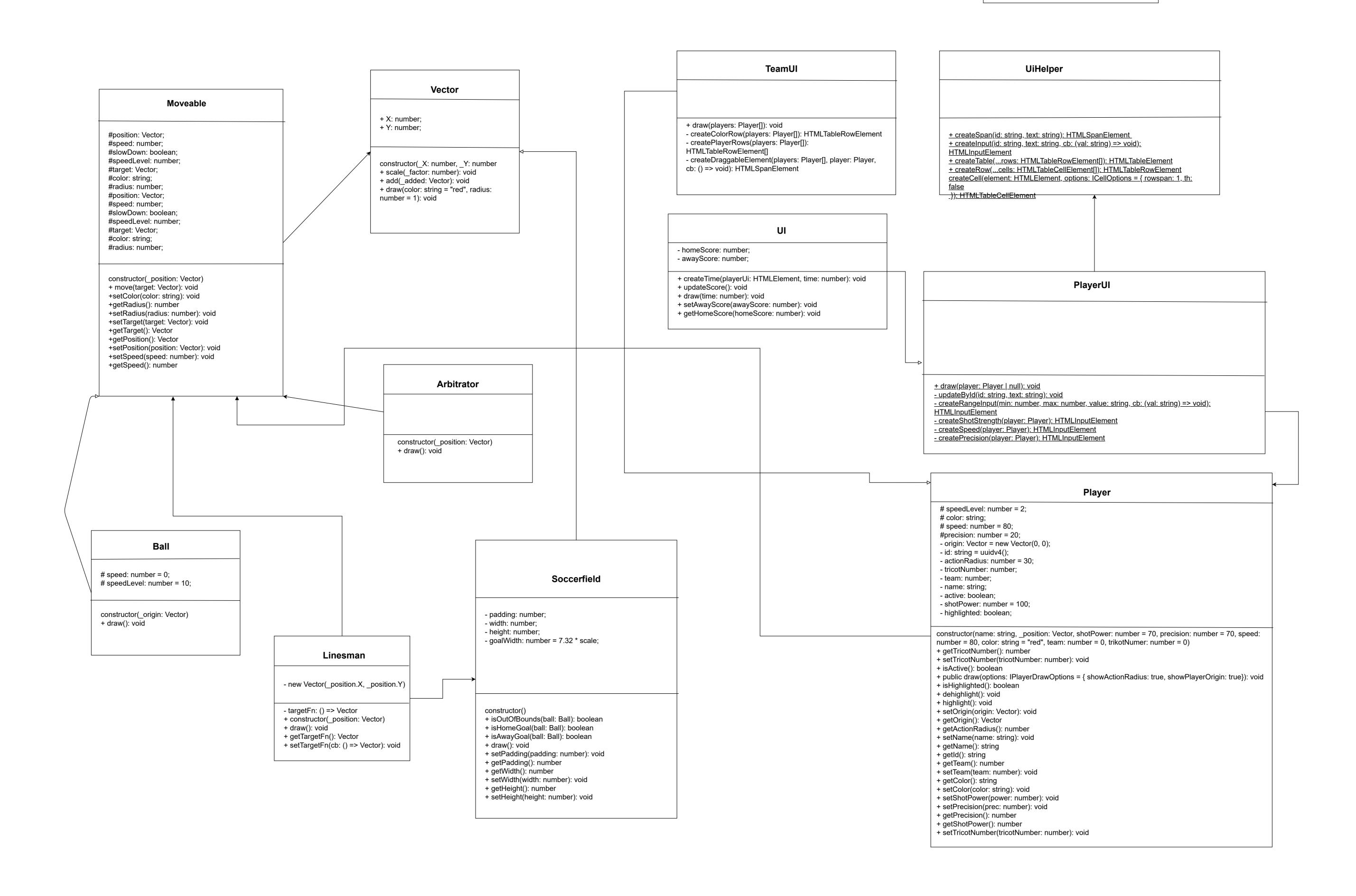
<< interface >>

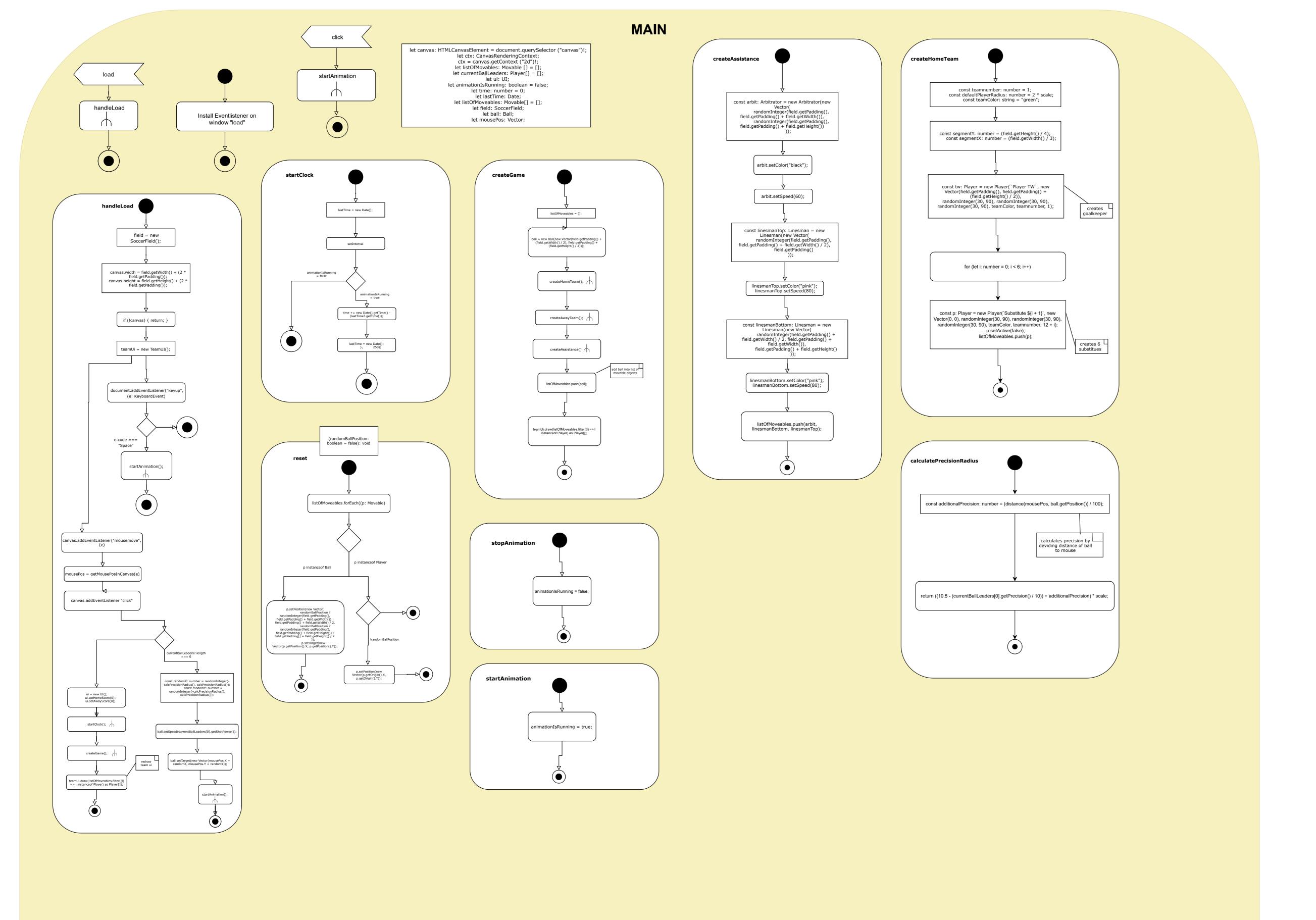
ICellOptions

rowspan?: number; th?: boolean;



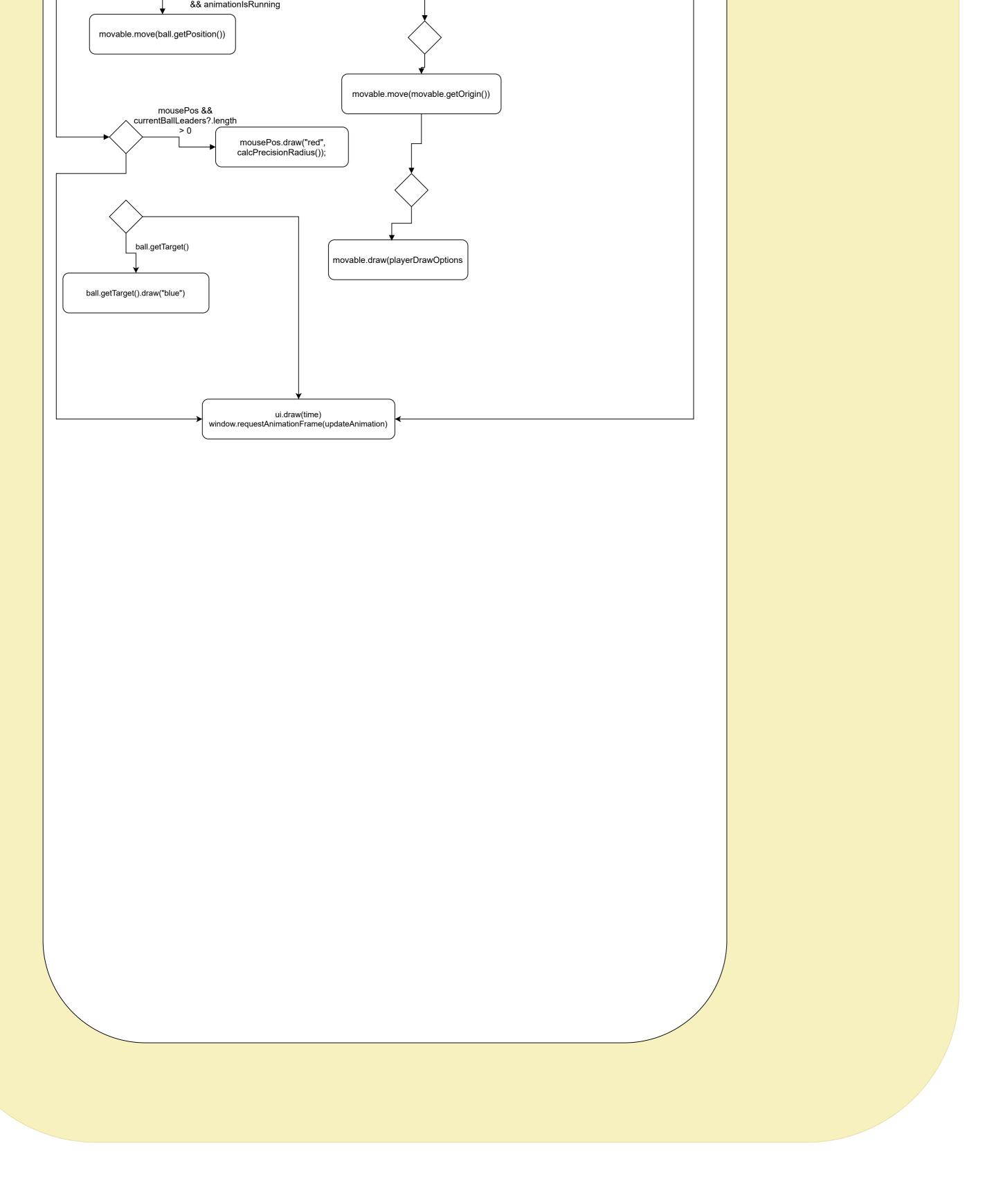
showActionRadius?: boolean; showPlayerOrigin?: boolean;

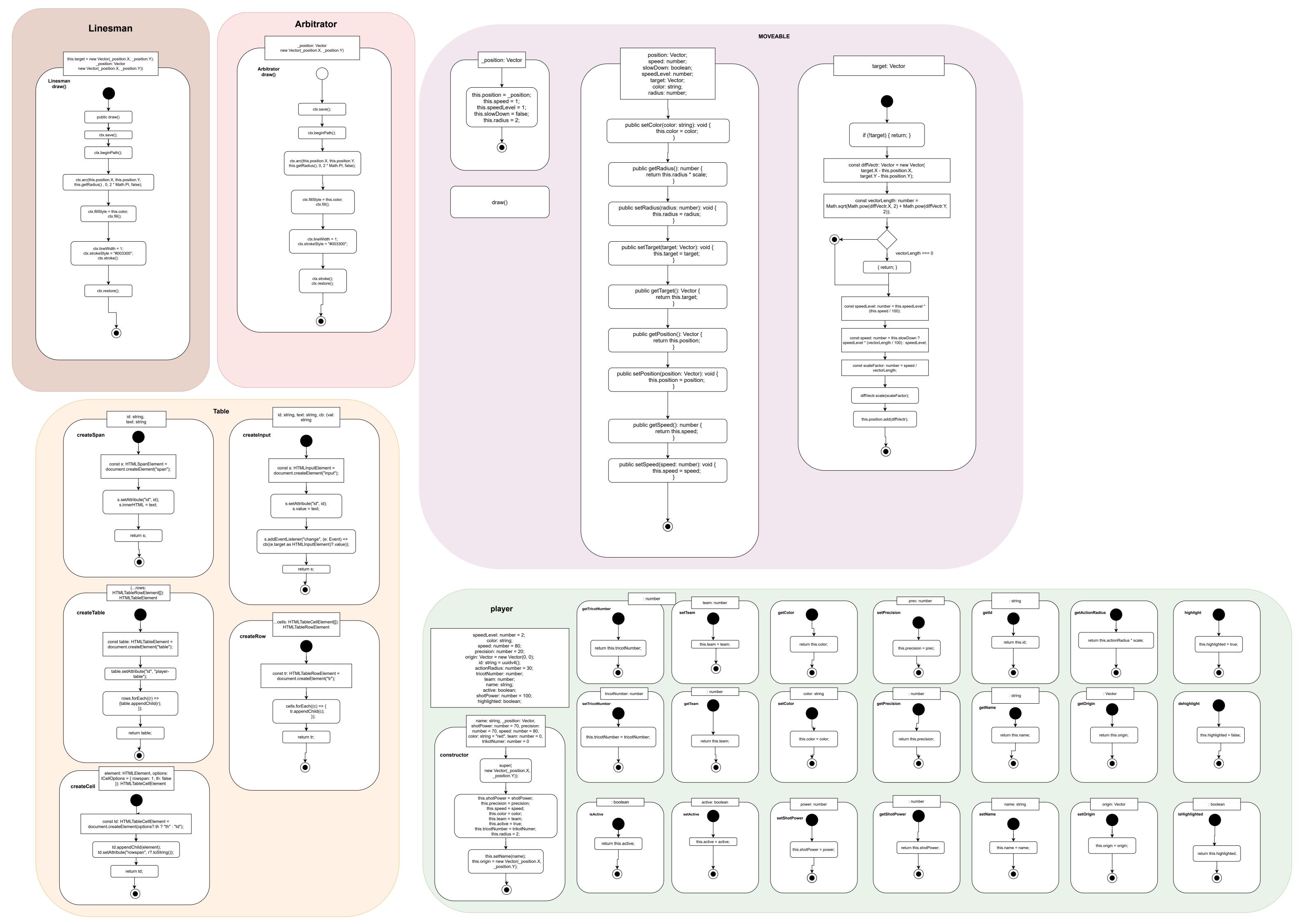


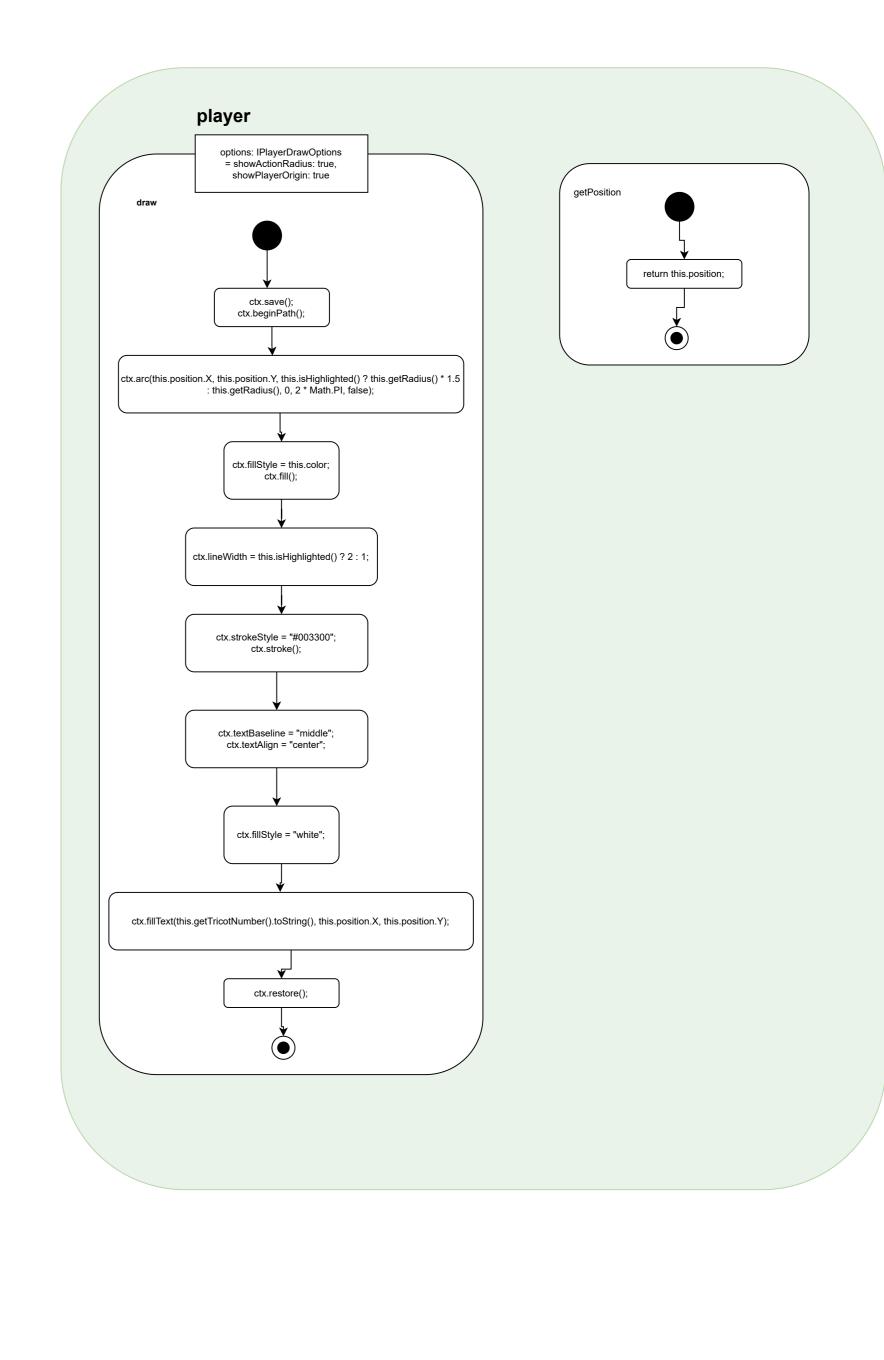


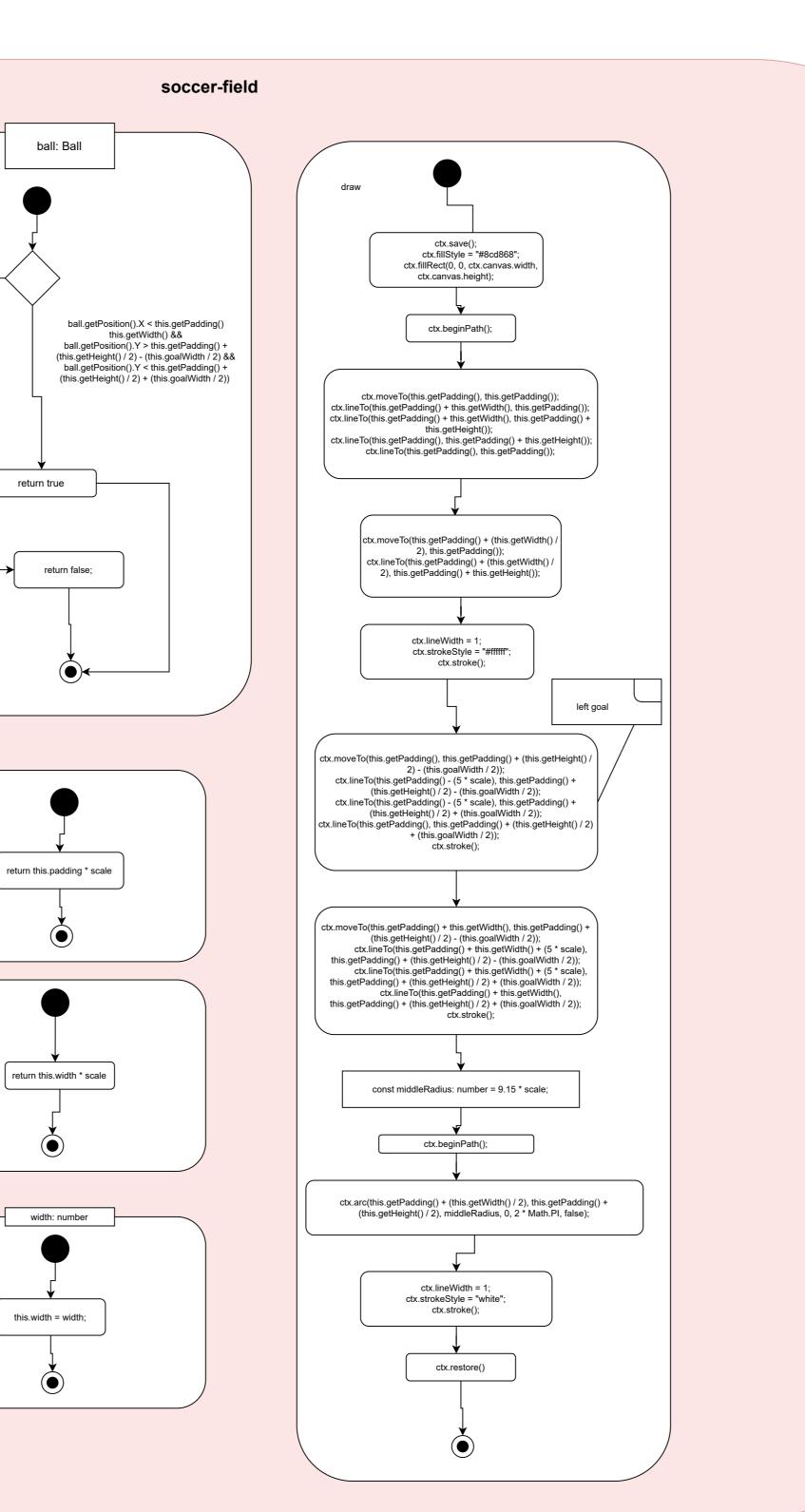
MAIN field.draw(); currentBallLeaders.length === 0 { PlayerUI.draw(null); } for (let movable of listOfMoveables) movable instanceof Arbitrator movable instanceof Linesman animationIsRunning movable.move(movable.getTargetFn()); } movable.draw(); randomInteger(0, 100) > 95 movable.setTarget(new Vector(randomInteger(field.getPadding(), field.getPadding() + field.getWidth()), randomInteger(field.getPadding(), field.getPadding() + field.getHeight()) ball.getTarget() && animationIsRunning movable instanceof const d: number = distance(movable.getPosition(), ball.getTarget()) movable instanceof Player const d: number =
distance(movable.getPosition(),
ball.getPosition()) - movable.getRadius() ball.getRadius();
const p: Player = movable; movable.move(ball.getTarget()) const isBallLeader: number = currentBallLeaders.findIndex((I) => I.getId() === p.getld()); field.isAwayGoal(movable)) movable.draw(); isBallLeader >= 0 && d > 0currentBallLeaders.splice(isBallLeader); ui.setHomeScore(ui.getHomeScore() + 1); stopAnimation(); reset(); d <= 0 && isBallLeader === -1 field.isHomeGoal(movable ui.setAwayScore(ui.getAwayScore() + 1) stopAnimation() reset() (field.isOutOfBounds(movable)) stopAnimation() reset(true) ` currentBallLeaders.push(movable); → stopAnimation(); PlayerUI.draw(movable); d <= movable.getActionRadius()











ball: Ball

return true

return true

return true

return true

return this.height * scale

ball.getPosition().Y < this.getPadding()

ball.getPosition().X <

this.getPadding()

isAwayGoal

getPadding

getWidth

setWidth

isOutOfBounds

ball.getPosition().X >
this.getPadding() +
this.getWidth()

padding: number;

width: number; height: number; goalWidth: number = 7.32 * scale;

this.setWidth(100);

this.setHeight(75); this.setPadding(10);

ball: Ball

return true

return false;

ball.getPosition().X < this.getPadding() && ball.getPosition().Y > this.getPadding() + (this.getHeight() / 2) - (this.goalWidth / 2) && ball.getPosition().Y < this.getPadding() +

(this.getHeight() / 2) + (this.goalWidth / 2))

this.height = height;

constructor

isHomeGoal

