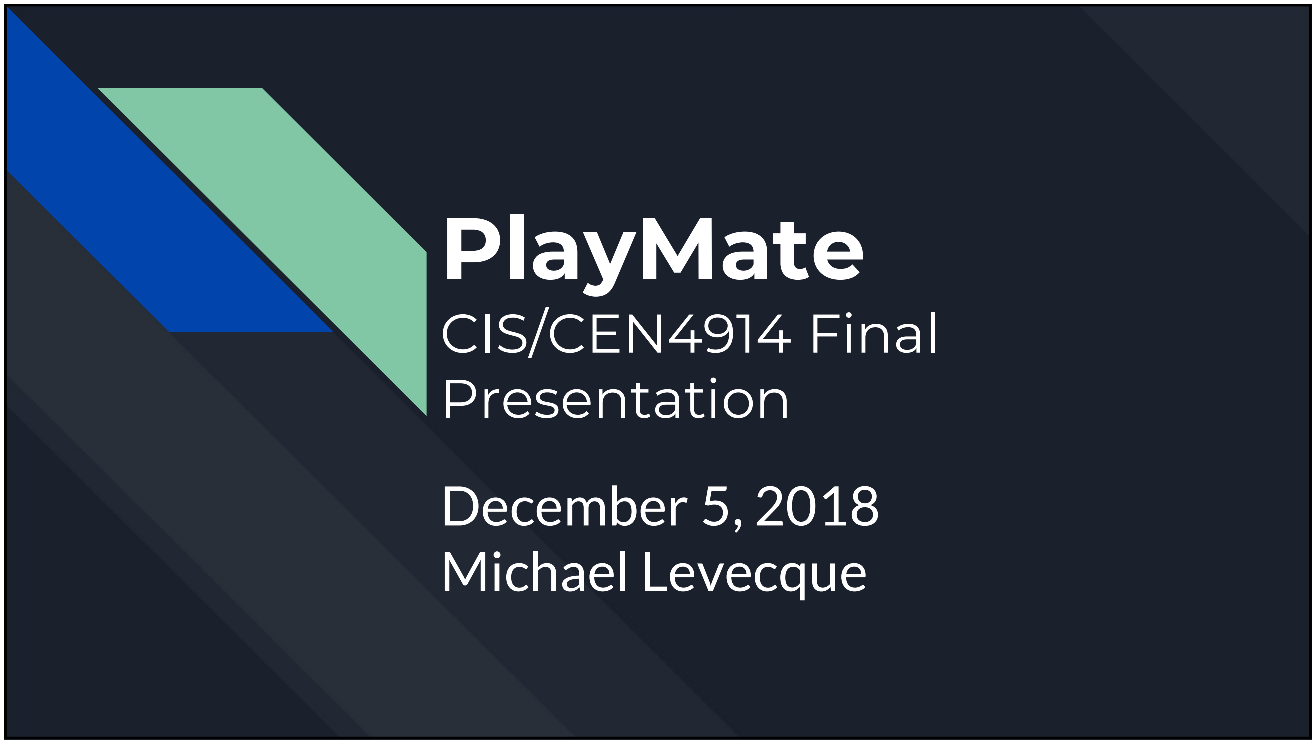
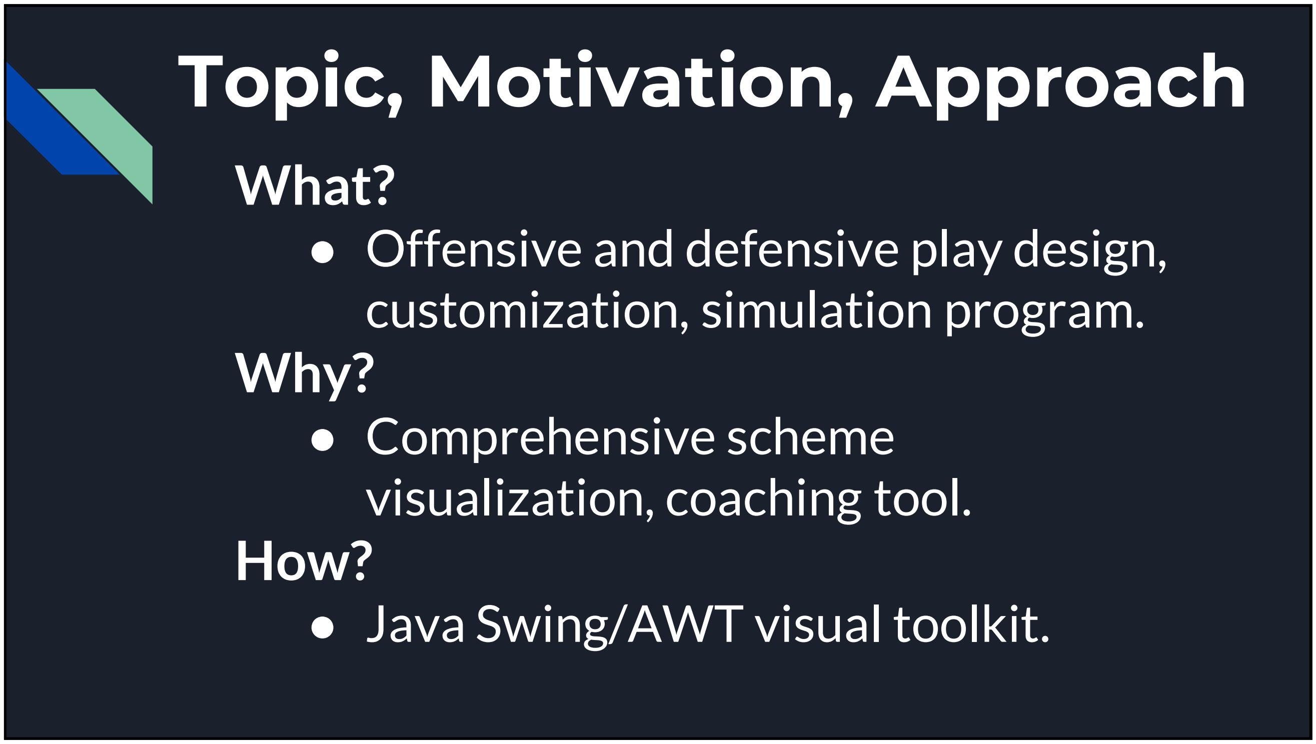


PlayMate

CIS/CEN4914 Final Presentation

December 5, 2018
Michael Levecque

1



Topic, Motivation, Approach

What?

- Offensive and defensive play design, customization, simulation program.

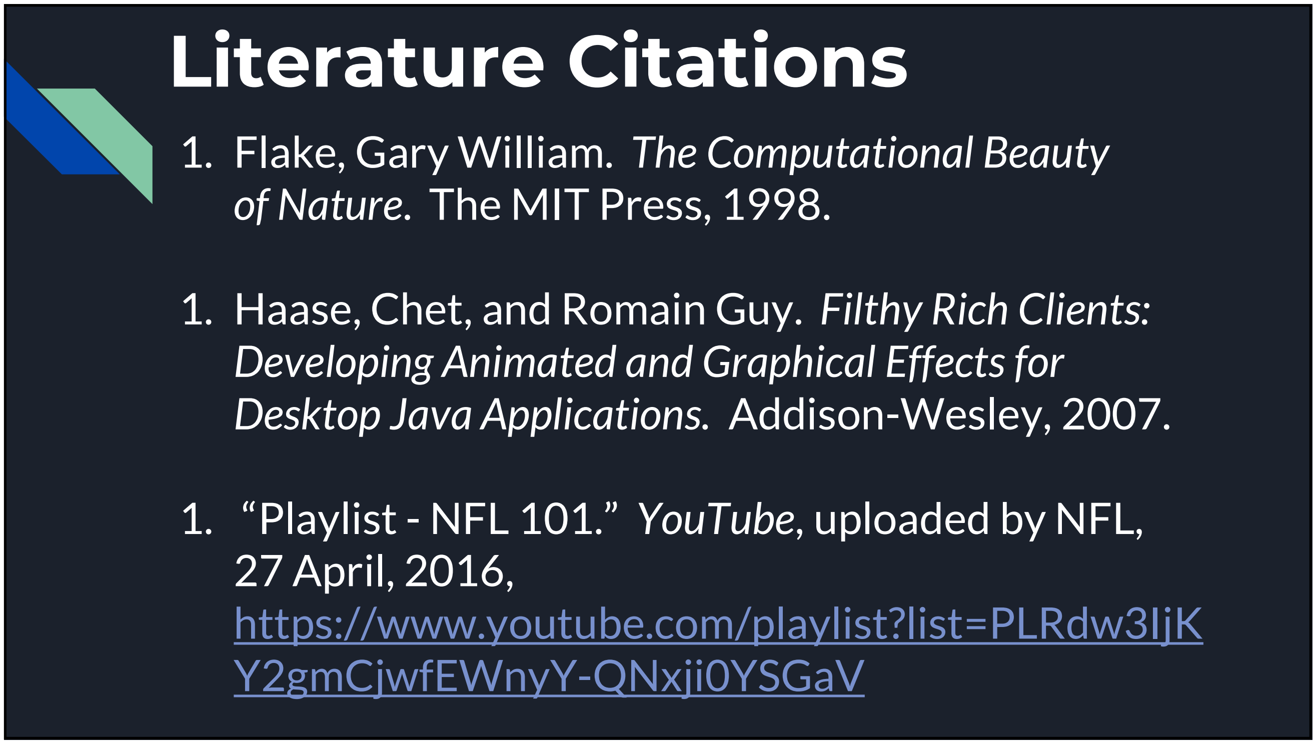
Why?

- Comprehensive scheme visualization, coaching tool.

How?

- Java Swing/AWT visual toolkit.

2



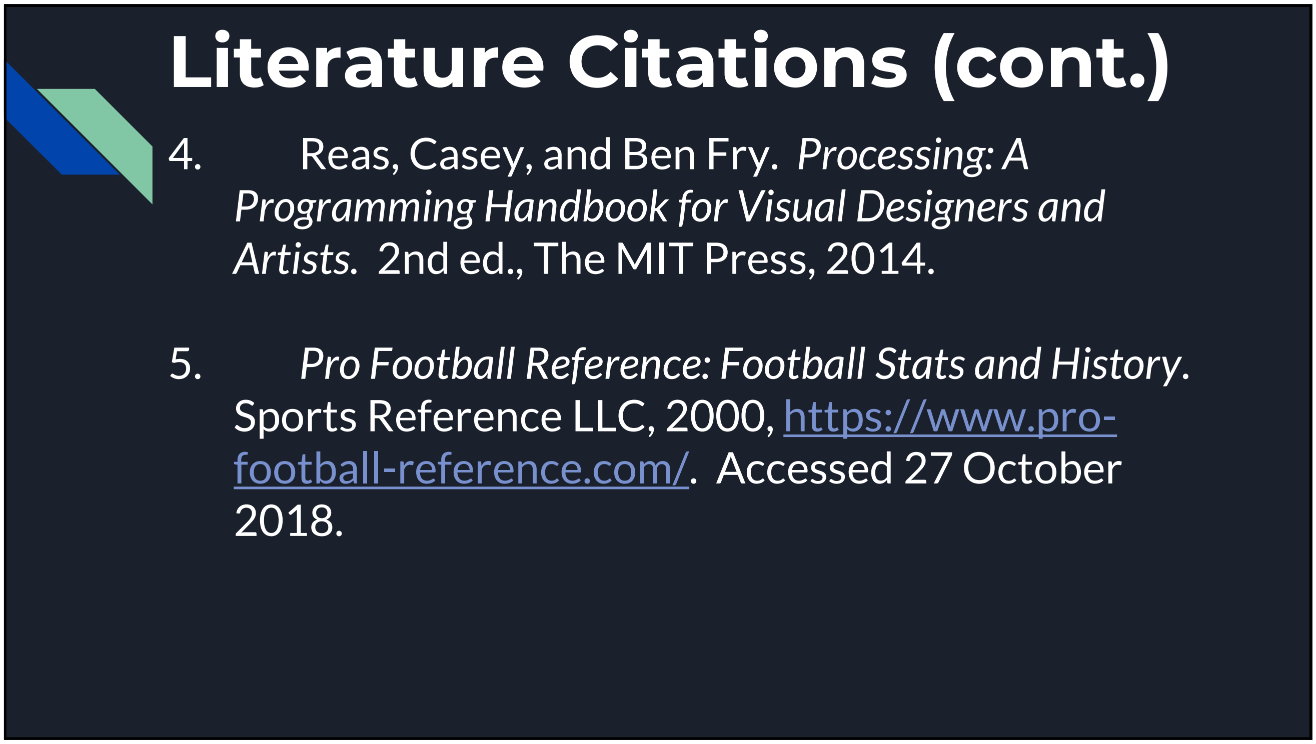
Literature Citations

1. Flake, Gary William. *The Computational Beauty of Nature.* The MIT Press, 1998.

1. Haase, Chet, and Romain Guy. *Filthy Rich Clients: Developing Animated and Graphical Effects for Desktop Java Applications.* Addison-Wesley, 2007.

1. "Playlist - NFL 101." *YouTube*, uploaded by NFL, 27 April, 2016, https://www.youtube.com/playlist?list=PLRdw3IjKY2gmCjwfEWnyY-QNxji0YSGaV

3



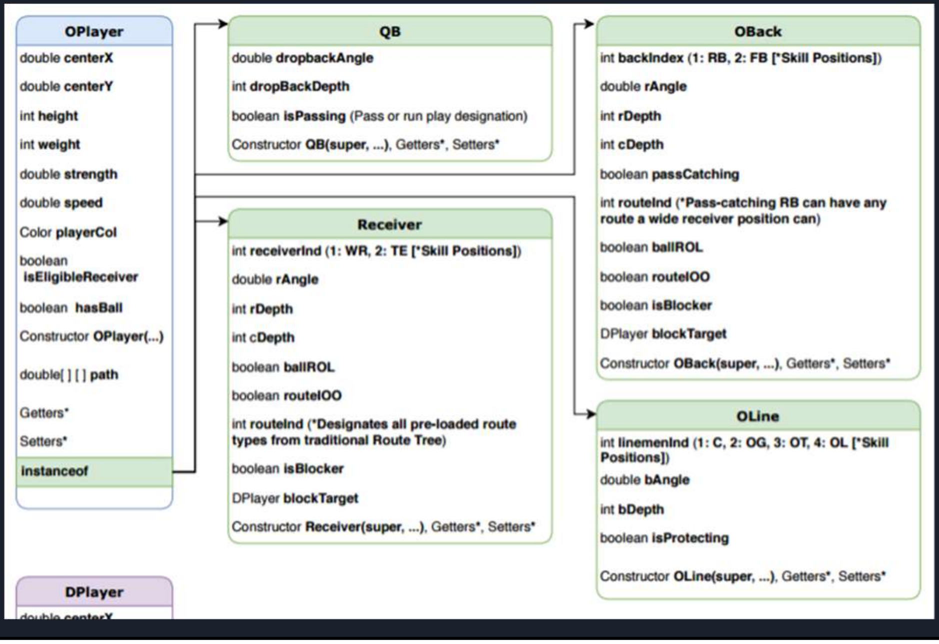
Literature Citations (cont.)

4. Reas, Casey, and Ben Fry. *Processing: A Programming Handbook for Visual Designers and Artists.* 2nd ed., The MIT Press, 2014.

5. *Pro Football Reference: Football Stats and History.* Sports Reference LLC, 2000, https://www.pro-football-reference.com/. Accessed 27 October 2018.

4

Custom Player Class Diagram



OPlayer
- double centerX
- double centerY
- int height
- int weight
- double strength
- double speed
- Color playerCol
- boolean isEligibleReceiver
- boolean hasBall
- Constructor OPlayer(...)
- double[ ] [ ] path
- Getters*
- Setters*
- instanceof

QB
- double dropbackAngle
- int dropBackDepth
- boolean isPassing (Pass or run play designation)
- Constructor QB(super, ...), Getters*, Setters*

Receiver
- int receiverInd (1: WR, 2: TE [*Skill Positions])
- double rAngle
- int rDepth
- int cDepth
- boolean ballROL
- boolean routeIOO
- int routeInd (*Designates all pre-loaded route types from traditional Route Tree)
- boolean isBlocker
- DPlayer blockTarget
- Constructor Receiver(super, ...), Getters*, Setters*

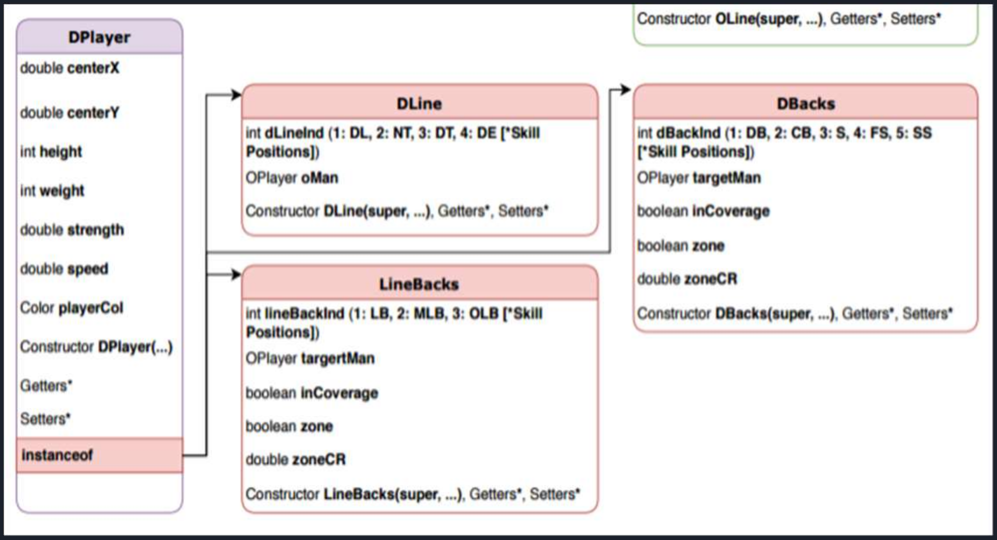
OBack
- int backIndex (1: RB, 2: FB [*Skill Positions])
- double rAngle
- int rDepth
- int cDepth
- boolean passCatching
- int routeInd (*Pass-catching RB can have any route a wide receiver position can)
- boolean ballROL
- boolean routeIOO
- boolean isBlocker
- DPlayer blockTarget
- Constructor OBack(super, ...), Getters*, Setters*

OLine
- int linemenInd (1: C, 2: OG, 3: OT, 4: OL [*Skill Positions])
- double bAngle
- int bDepth
- boolean isProtecting
- Constructor OLine(super, ...), Getters*, Setters*

DPlayer
- double centerX

5

Class Diagram (cont.)



- Constructor OLine(super, ...), Getters*, Setters*

DPlayer
- double centerX
- double centerY
- int height
- int weight
- double strength
- double speed
- Color playerCol
- Constructor DPlayer(...)
- Getters*
- Setters*
- instanceof

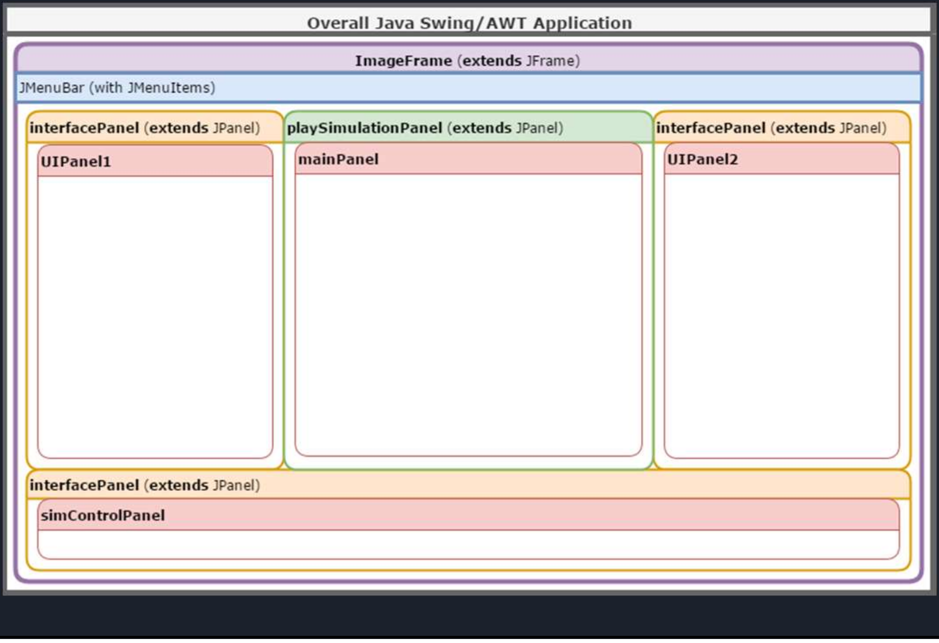
DLine
- int dLineInd (1: DL, 2: NT, 3: DT, 4: DE [*Skill Positions])
- OPlayer oMan
- Constructor DLine(super, ...), Getters*, Setters*

LineBacks
- int lineBackInd (1: LB, 2: MLB, 3: OLB [*Skill Positions])
- OPlayer targertMan
- boolean inCoverage
- boolean zone
- double zoneCR
- Constructor LineBacks(super, ...), Getters*, Setters*

DBacks
- int dBackInd (1: DB, 2: CB, 3: S, 4: FS, 5: SS [*Skill Positions])
- OPlayer targetMan
- boolean inCoverage
- boolean zone
- double zoneCR
- Constructor DBacks(super, ...), Getters*, Setters*

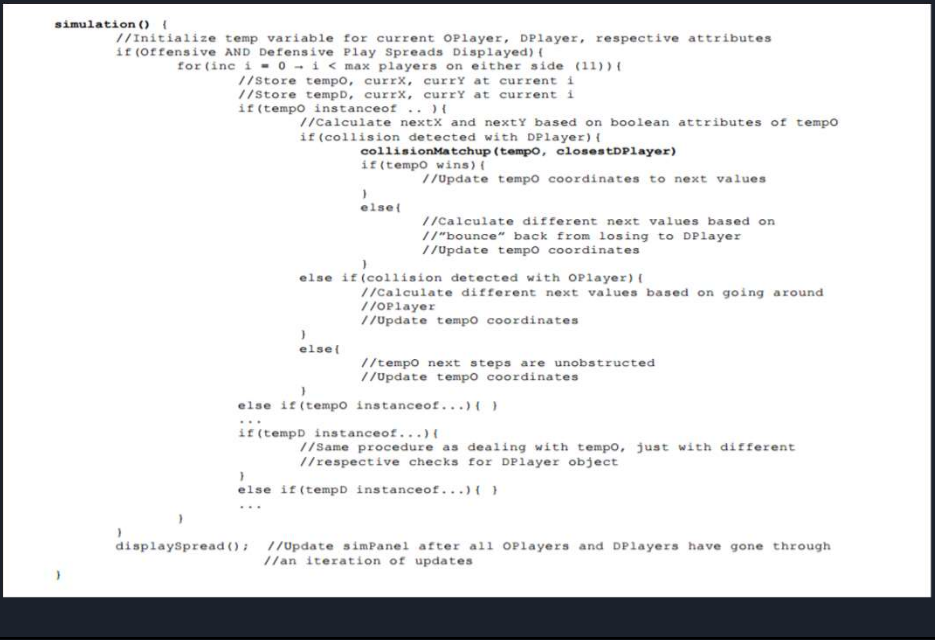
6

System Structure


Overall Java Swing/AWT Application
ImageFrame (extends JFrame)
JMenuBar (with JMenuItems)
interfacePanel (extends JPanel)
UIPanel1
playSimulationPanel (extends JPanel)
mainPanel
interfacePanel (extends JPanel)
UIPanel2
interfacePanel (extends JPanel)
simControlPanel

7

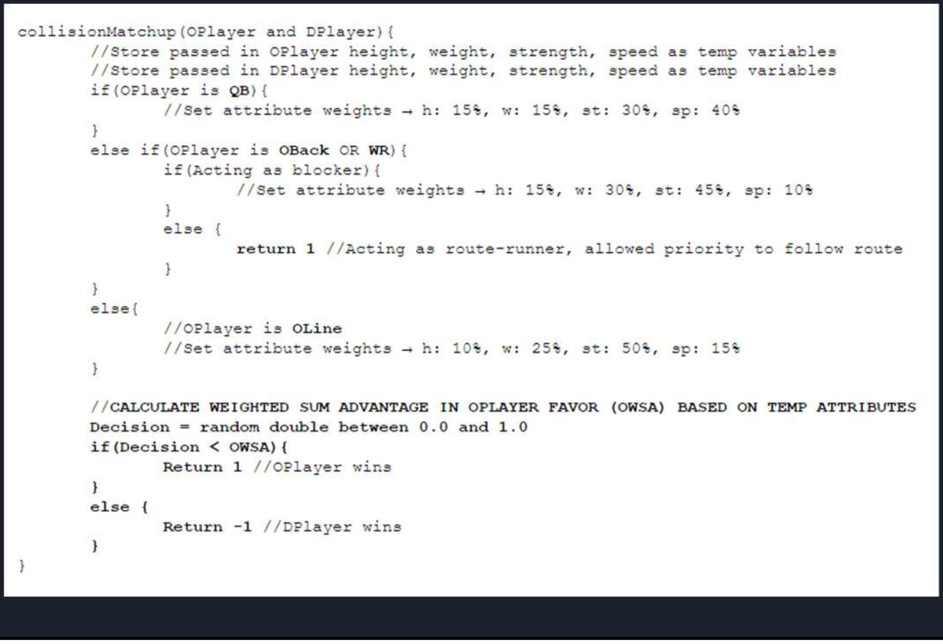
Simulation



```
simulation() {
        //Initialize temp variable for current OPlayer, DPlayer, respective attributes
        if(Offensive AND Defensive Play Spreads Displayed){
                for(inc i = 0 → i < max players on either side (11)){
                        //Store tempO, currX, currY at current i
                        //Store tempD, currX, currY at current i
                        if(tempO instanceof .. ){
                                //Calculate nextX and nextY based on boolean attributes of tempO
                                if(collision detected with DPlayer){
                                        collisionMatchup(tempO, closestDPlayer)
                                        if(tempO wins){
                                                //Update tempO coordinates to next values
                                        }
                                        else{
                                                //Calculate different next values based on
                                                //"bounce" back from losing to DPlayer
                                                //Update tempO coordinates
                                        }
                                else if(collision detected with OPlayer){
                                        //Calculate different next values based on going around
                                        //OPlayer
                                        //Update tempO coordinates
                                }
                                else{
                                        //tempO next steps are unobstructed
                                        //Update tempO coordinates
                                }
                        else if(tempO instanceof...){ }
                        ...
                        if(tempD instanceof...){
                                //Same procedure as dealing with tempO, just with different
                                //respective checks for DPlayer object
                        }
                        else if(tempD instanceof...){ }
                        ...
                }
        }
        displaySpread();  //Update simPanel after all OPlayers and DPlayers have gone through
                          //an iteration of updates
}
```

8

Collision Matchup



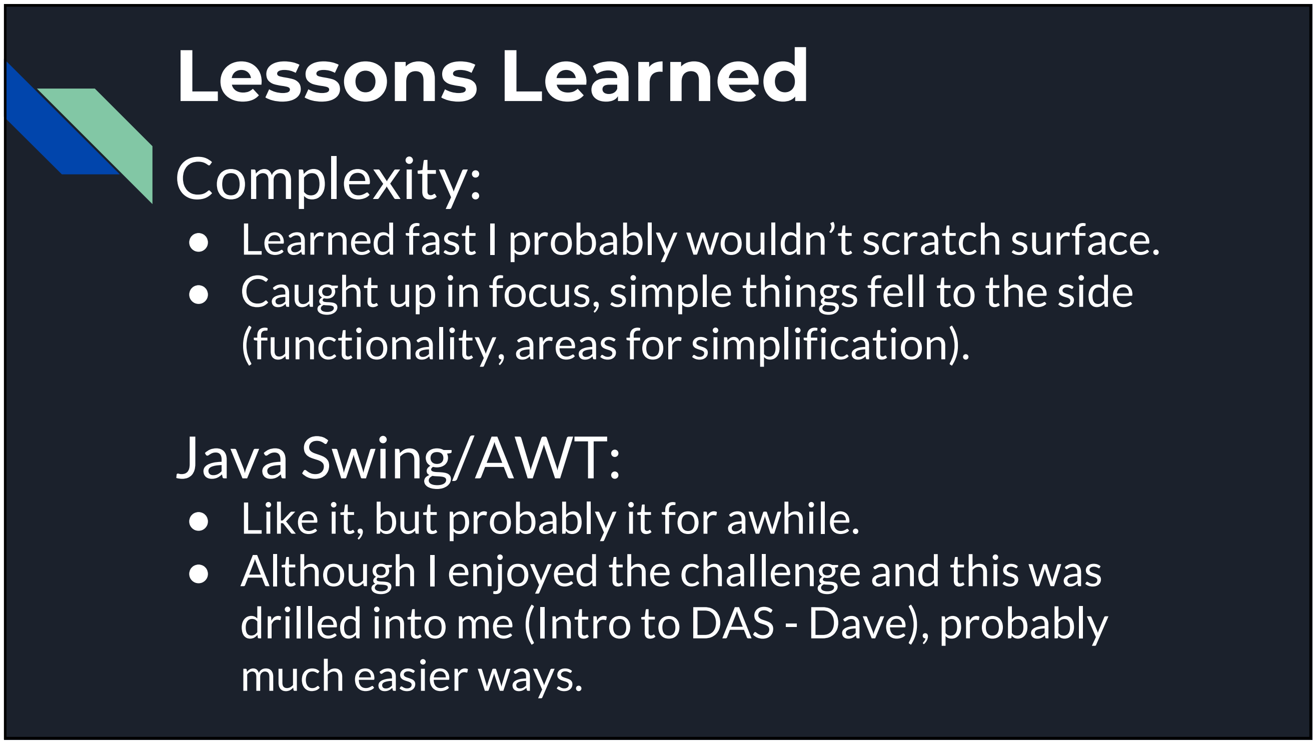
```
collisionMatchup(OPlayer and DPlayer){
        //Store passed in OPlayer height, weight, strength, speed as temp variables
        //Store passed in DPlayer height, weight, strength, speed as temp variables
        if(OPlayer is QB){
                //Set attribute weights → h: 15%, w: 15%, st: 30%, sp: 40%
        }
        else if(OPlayer is OBack OR WR){
                if(Acting as blocker){
                        //Set attribute weights → h: 15%, w: 30%, st: 45%, sp: 10%
                }
                else {
                        return 1 //Acting as route-runner, allowed priority to follow route
                }
        }
        else{
                //OPlayer is OLine
                //Set attribute weights → h: 10%, w: 25%, st: 50%, sp: 15%
        }

        //CALCULATE WEIGHTED SUM ADVANTAGE IN OPLAYER FAVOR (OWSA) BASED ON TEMP ATTRIBUTES
        Decision = random double between 0.0 and 1.0
        if(Decision < OWSA){
                Return 1 //OPlayer wins
        }
        else {
                Return -1 //DPlayer wins
        }
}
```

9

DEMO

10



Lessons Learned

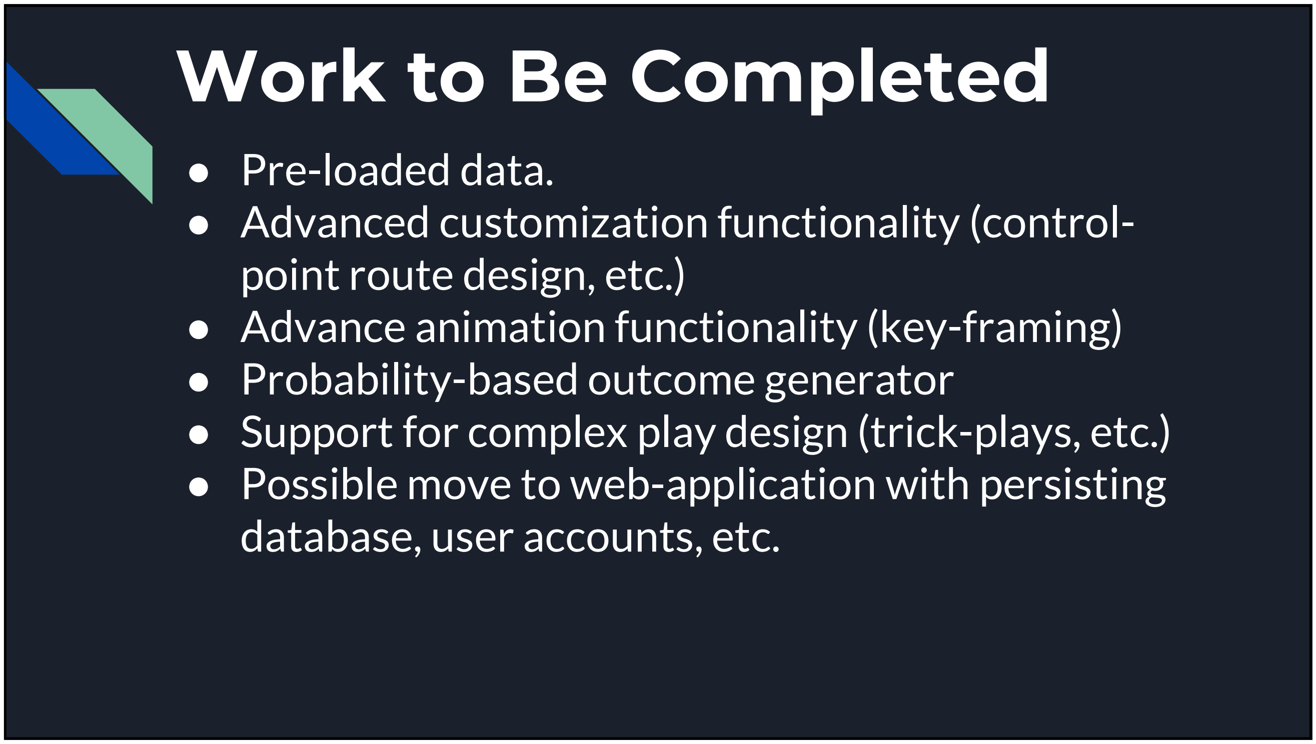
Complexity:

- Learned fast I probably wouldn’t scratch surface.
- Caught up in focus, simple things fell to the side (functionality, areas for simplification).

Java Swing/AWT:

- Like it, but probably it for awhile.
- Although I enjoyed the challenge and this was drilled into me (Intro to DAS - Dave), probably much easier ways.

11



Work to Be Completed

- Pre-loaded data.
- Advanced customization functionality (control-point route design, etc.)
- Advance animation functionality (key-framing)
- Probability-based outcome generator
- Support for complex play design (trick-plays, etc.)
- Possible move to web-application with persisting database, user accounts, etc.

12

QUESTIONS?

13