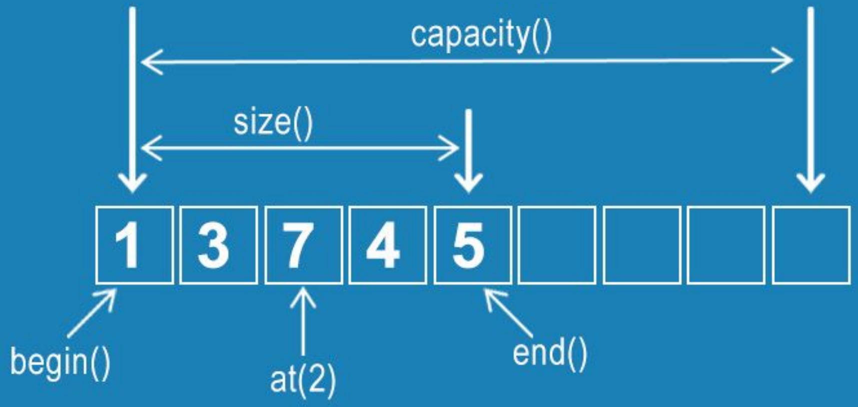
Data Structures in our Arcade Cabinet!

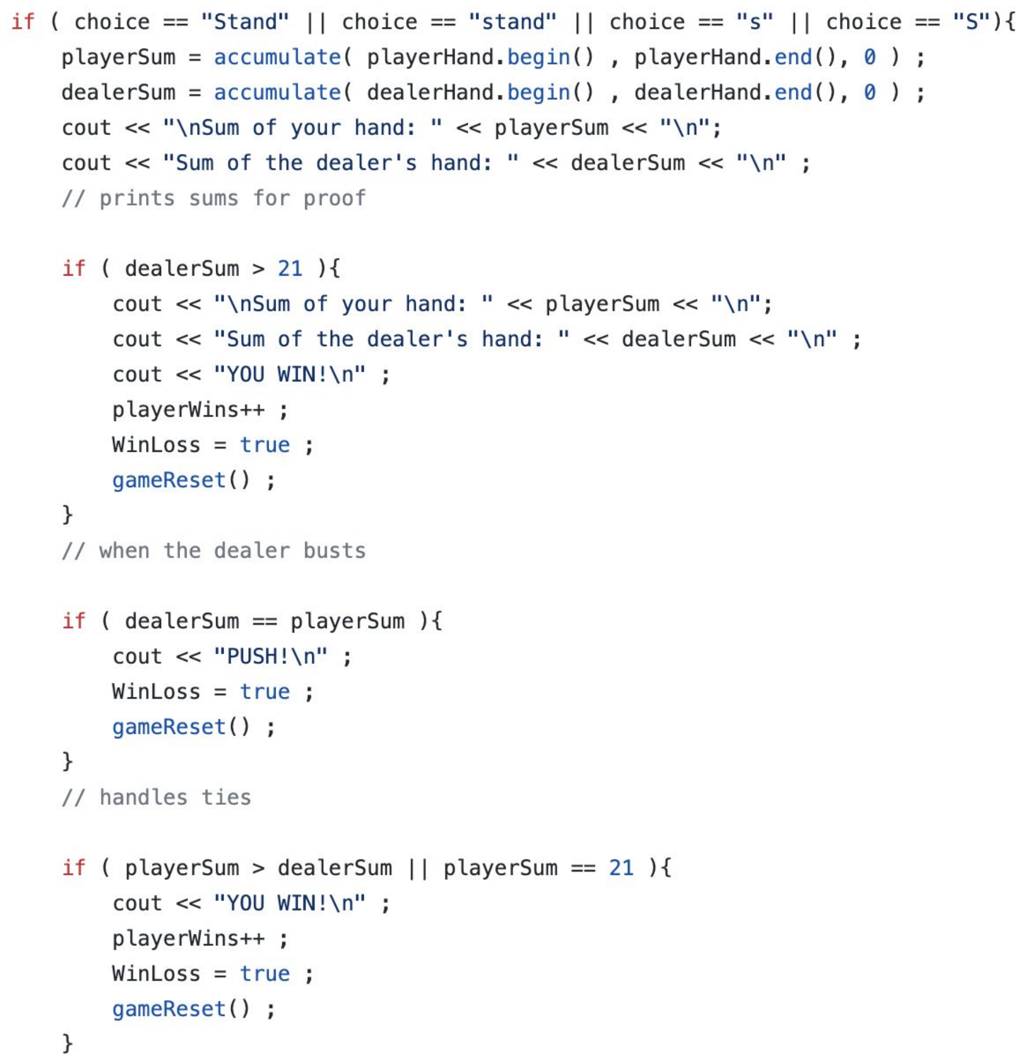
Carter Earheart-Brown,
Stephanie Hill, Swasti Mishra, Katie Nuchols

Qualities of Vectors

- Easy to resize them with insertion and deletion
- No automatic sorting
- Easy to add up all of the elements
- They are usually intended to be accessed in order
    - I.e., it takes more time to access elements that are further down the line

Vector Structure



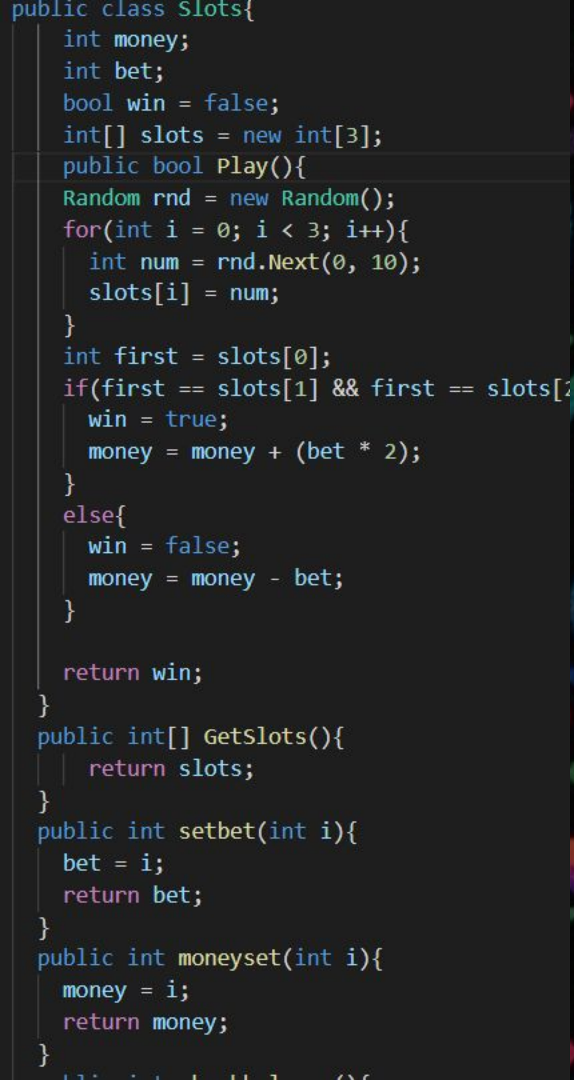


```
if ( choice == "Stand" || choice == "stand" || choice == "s" || choice == "S"){
    playerSum = accumulate( playerHand.begin() , playerHand.end(), 0 ) ;
    dealerSum = accumulate( dealerHand.begin() , dealerHand.end(), 0 ) ;
    cout << "\nSum of your hand: " << playerSum << "\n";
    cout << "Sum of the dealer's hand: " << dealerSum << "\n" ;
    // prints sums for proof

    if ( dealerSum > 21 ){
        cout << "\nSum of your hand: " << playerSum << "\n";
        cout << "Sum of the dealer's hand: " << dealerSum << "\n" ;
        cout << "YOU WIN!\n" ;
        playerWins++ ;
        WinLoss = true ;
        gameReset() ;
    }
    // when the dealer busts

    if ( dealerSum == playerSum ){
        cout << "PUSH!\n" ;
        WinLoss = true ;
        gameReset() ;
    }
    // handles ties

    if ( playerSum > dealerSum || playerSum == 21 ){
        cout << "YOU WIN!\n" ;
        playerWins++ ;
        WinLoss = true ;
        gameReset() ;
    }
```

Vectors in BlackJack

- Resize them every time we push back a new card
- Sorting would not be preferred behavior
- Simple to add all cards in a player or dealer’s hand
- Intended to be accessed in order

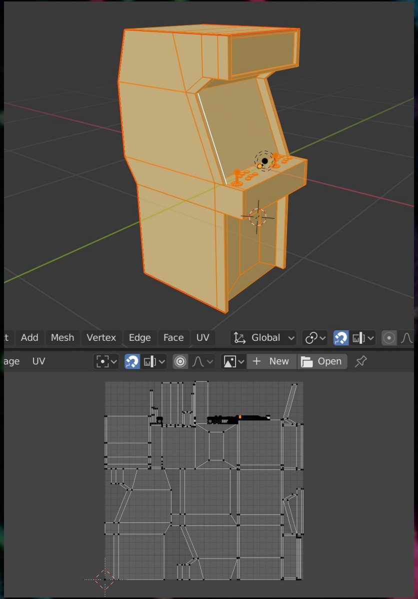
Vectors in Slot Machine

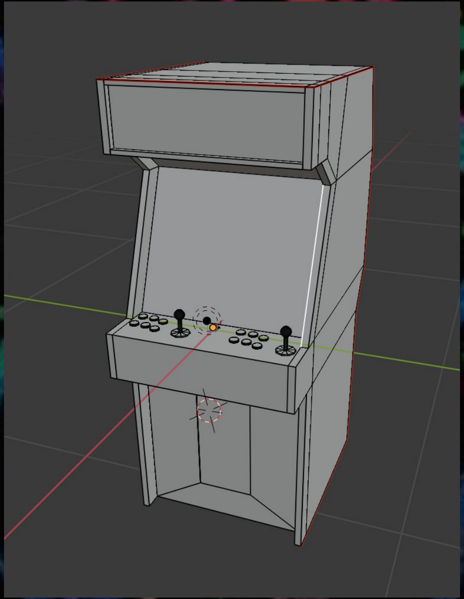


```
public class Slots{
    int money;
    int bet;
    bool win = false;
    int[] slots = new int[3];
    public bool Play(){
    Random rnd = new Random();
    for(int i = 0; i < 3; i++){
      int num = rnd.Next(0, 10);
      slots[i] = num;
    }
    int first = slots[0];
    if(first == slots[1] && first == slots[2
      win = true;
      money = money + (bet * 2);
    }
    else{
      win = false;
      money = money - bet;
    }

    return win;
  }
  public int[] GetSlots(){
      return slots;
  }
  public int setbet(int i){
    bet = i;
    return bet;
  }
  public int moneyset(int i){
    money = i;
    return money;
  }
```

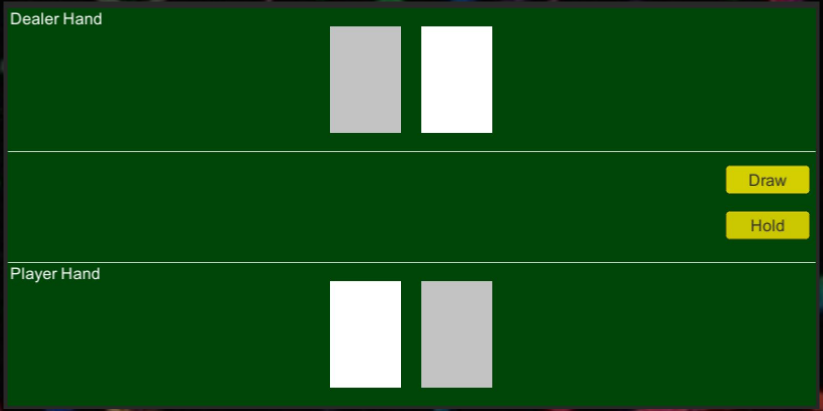
- Resize them every time a player both makes a bet and wins money
- Again, sorting would not be preferred behavior
- Totaling the amount of money a player has won or lost is extremely easy
- Intended to be accessed in order, basically as a history of the player’s activities

Putting It All Together!





- Converting the games from code to games in Unity
- Overlaying the games onto the arcade cabinet

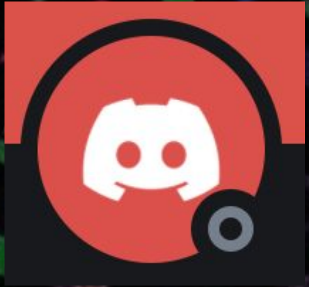


Team Members

Carter Earheart-Brown

Swasti Mishra

Stephanie Hill



Katie Nuchols