# Pigs

Void pigs(void) är huvudfunktionen för det första spelet. Här använde vi oss av ett liknande upplägg som "craps" utgått ifrån, en tydlig grund för programmet som följer för att få spelet att fungera. För att starta spelet kallar den på funktionen playPigsGame, funktionen frågar även efter om användaren vill ha instruktioner, ifall så är fallet så kallar den på funktionen pigsInstructions (funktionen presenterar instruktioner). Funktionen playPigsGame är även en knutpunkt vid förlorad eller vunnen spelomgång, efter avslutad spelomgång frågar funktionen om användaren vill spela igen, ifall svaret är ja så börjar en ny spelomgång, annars återgår programmet till huvudmenyn (main funktionen). Funktionen playPigsGame sköter i stort sett turordningen för användaren och datorn via en while-loop. I funktionen lagras det totala värdet som har vunnits av bägge parterna i deras respektive turer (humanTurn, computerTurn). När användaren spelar så kallar funktionen playPigsGame på funktionen humanTurn, vilket representerar användarens tur att spela. I humanTurn används funktionen rollOneDice som slumpar fram ett heltal mellan 1-6 och returnerar ett värde till playPigsGame. Efter användarens respektive datorns så omgång kontrolleras resultatet, om det överskrider eller är lika med 100 så avslutas spelet. Programmet återgår då till funktionen huvudfunktionen pigs, som då frågar användaren om han/hon vill spela igen, eller att återgå till huvudmenyn.
Samma princip följer datorns tur, bara att den kallar på funktionen computerTurn vilket representerar datorns tur. Datorn kan maximalt kasta tärningarna tre gånger per tur. Efter två lyckade tärningskast (med resultat över 1) slumpas ifall en tredje omgång skall slås (50% chans). Detta via att kalla på funktionen computerExtraRoll. Vardera turer (användaren och datorns) tar som input det totala värdet som lagrats i playPigsGame i två olika variabler och returnerar det nya värdet från bägges omgångar tillbaka till playPigsGame, där det nya värdet från dess omgångar adderas till de lagrade variablerna (totalvariablerna) hos bägge parterna fram tills dess att någon har nått till 100 eller över.

Vi valde att dela upp spelet i dessa olika funktioner för att skapa ett tydligt upplägg. Därav valde vi att skriva separata funktioner för datorns tur och användarens tur som tillkallas i en huvudfunktion, detta dels p.g.a. användarens tur var baserat på val (input) från användaren medan datorns tur var baserad utifrån en tydlig och rak strategi (förutom 50/50 chans att den kastar ett tredje tärningskast). Vi började med att skriva koden till detta spel utifrån "människans" tur, men märkte snabbt att vi fastnat på att lägga mycket tid på smådetaljer. Därför valde vi att börja bygga programmet från en slags "brygga", vilket ledde till playPigsGame-funktionen, som fungerar som en slags turfördelare mellan användarens och datorns tur. Vi hade också till en början svårt att lagra det totala värdet när vi inriktade oss mer på detaljer som start. Detta blev enklare då vi kunde lagra de totala variablerna för användaren och datorn i "bryggan" (playPigsGame).

# Two Dice Pigs

Void playTwoDicePigs(void) är huvudfunktionen för det andra spelet. Här utgick vi mycket ifrån samma struktur och uppdelning av funktioner som vi använt oss av i pigs. Funktionen playTwoDicePigs skiljer sig inte mycket ifrån motsvarande playPigsGame-funktionen i pigs. Skillnaden är egentligen att spelet spelas med två tärningar och har lite annorlunda regler. I twoDicePigsHumanTurn, twoDicePigsComputerTurns samt twoDicePigsComputerExtraRoll så finns det fler if satser än i vanliga pigs. Detta beror på de regler som gäller i twoDicePigs, de nya reglerna sätter nya villkor. Blir en av tärningarna en 1:a så returneras 0 till totala poängen och turen går över till nästa spelaren. Funktionerna kan också returnera värdet (-1) till twoDicePigs ifall någon av funktionerna har fått två stycken ettor i sina tärningskast. Returneras (-1) så nollställs human/computers totala poäng helt och turen går över till nästa spelaren.

Vi valde att följa vår tidigare struktur i pigs och återanvände mycket av koden. Vi valde att använda oss av funktionen rollOneDice två gånger i twoDicePigsHumanTurn, twoDiceComputerTurn och twoDiceComputerExtraRoll funktionerna för att få ut två separata värden och deklarera till två olika variabler, detta för att enkelt kunna jämföra dem med varann. Det fanns en regel som klassades som en möjlig “sub-variation” av spelet two dice pigs, denna regel var att ifall tärningarna visade samma siffra (inte en 1:a) så skulle användaren/datorn tvingas att kasta tärningarna igen. Vi valde att inte ha med denna alternativa regel i vårt spel. Till en början hade vi svårt med att få med i våra funktioner om användaren/datorn kastade två ettor att deras totala poäng skulle nollställas, eller ifall en av tärningarna visade en 1:a. Vi löste detta genom att returnera värdet (-1) i funktionerna twoDiceHumanTurn, twoDiceComputerTurn och twoDiceComputerExtraRoll i villkoret där utfallet av två kast blev två stycken ettor eller (0) ifall en av tärningarna var en 1:a. När värdena returnerats till playTwoDicePigs så nollställdes total vid (-1). Returnerades (0) var totalen oförändrad.

# Roulette

I roulette-spelet utgick vi från en liknande struktur som i kodandet av Pigs-spelen. Vi startade med en "main"-funktion kallad roulette, vid den blir man också frågad som i Pigs om instruktioner innan spelet startas. Efter det tillkallas en funktion vid namn rouletteChoice som i stort sett frågar användaren vilken typ av roulette han/hon vill spela, samt ifall användaren vill avsluta spelet, i det fallet går den tillbaka till funktionen innan (roulette) som i sin tur frågar om användaren är säker på att han/hon vill avsluta. Ifall användaren har följt alla steg för att spela spelet så tillkallas funktionen europeanRoulette eller americanRoulette från rouletteChoice, som i sin tur i steg 1 frågar användaren om ett belopp mellan 100$-10000$ att spela med. Ifall användaren matar in ett felaktigt belopp utanför ramarna av det angivna så hänvisas användaren att skriva in beloppet igen. Efter det frågas användaren om vilket/vilka nummer han/hon vill spela med i steg 2. Ifall användaren enbart vill satsa på ett nummer så hänvisas användaren att skriva in det två gånger. Vid steg 3 frågar även funktionen om vilket bet han/hon vill satsa på sitt/sina nummer. Ifall användaren skriver in ett bet som har ett värde över sitt totala spelbelopp så hänvisas användaren att skriva om uppgifterna från steg 2. Efter att alla uppgifter är korrekta så tillkallas funktionen playEuropeanRoulette respektive playAmericanRoulette (ifall användaren från början har valt att spela detta alternativ av roulette). Inne i dessa funktioner sker själva spelet (man kan se det som spelhjulet). Det som sker inuti dessa funktioner är att de returnerar ett värde baserat på hur det valda numret/nummer förhåller sig till vad "roulettehjulet" stannat på. Alltså ett slumpmässigt heltal mellan 0-36 i den europeiska versionen, respektive 0-37 i den amerikanska där 37 omvandlas till en extra 0:a. Ifall det inte förekommer någon match mellan det valda nummret/nummer och det hjulet har stannat på så returneras bettet tillbaka till funktionerna europeanRoulette respektive americanRoulette och subtraheras bort från det totala spelbeloppet. Vid match av ett nummer så returneras 17 gånger pengarna, då vid satsning på två olika nummer, samt 35 gånger pengarna vid satsning på ett nummer. Skillnaden i den amerikanska versionen i funktionen playAmericanRoulette är att vid satsning på 0, så gångras pengarna med 17 oavsett, då det finns en extra nolla på spelhjulet. Vid varje avklarad speltur på roulettehjulet så frågar funktionerna europeanRoulette, samt americanRoulette om man vill avsluta spelet eller fortsätta.

Det som vi kände var viktigt var att få en tydlig struktur, därför valde vi som i de två tidigare spelen att börja från top-down. Alltså inrikta oss på att få alla huvudkomponenter på plats i programmet, för att sedan utveckla detaljer. T.ex. att få till en funktionell switch mellan de två olika typerna av roulette-spelen (funktionen rouletteChoice), för att sedan bygga vidare på de två olika spelen. Efter det kom vi fram till att vi skulle ha en funktion till vardera spel som behandlar all typ av input-data innan spelet börjar (funktionen europeanRoulette respektive americanRoulette) som i sin tur ledde till själva spelomgångarna playEuropeanRoulette respektive playAmericanRoulette som behandlar denna inputen från användaren för att sedan returnera ett värde tillbaka till "input"-funktionerna.