First Steps to a Qt Application

Benson Muite

Table of Contents

1. Introduction

In this tutorial, you will learn how to make a command line Qt6 application. Development can be done locally on your machine or on a remote cloud server. You can later extend the application to use a graphical user interface.

2. Mancala Games

Mancala games are a class of two player count and capture games. They are wide spread, many variants in Africa. They are strategy board games. In the USA Kalah is the most common variant. Bao la Kiswahili (board of the Kiswahili), is very common on the East African coast. Thera are a number of Mancala implementations available for computers and mobile phones, but there is a need to develop better automated players since two person network games are not very popular.

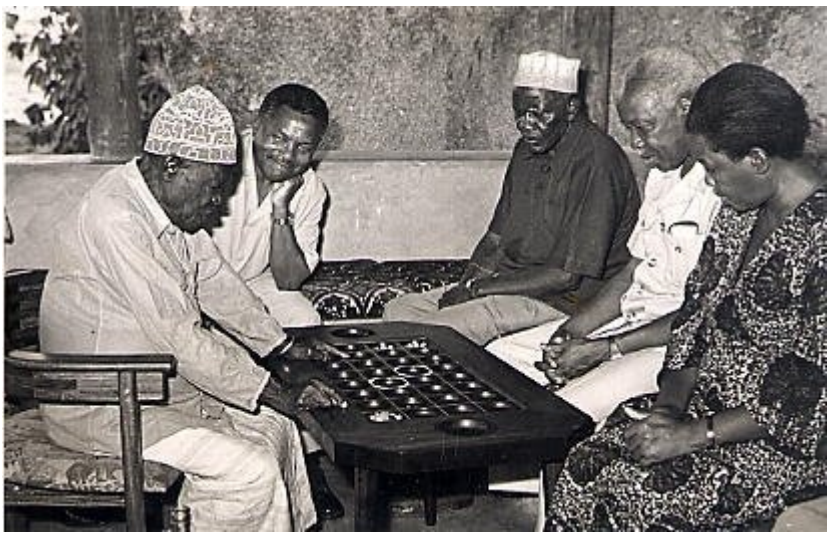

Figure 1. People playing Bao la Kiswahili, public domain image

2.1. Kalah

We will consider the rules given by Irving, Donkers and Uiterwijk, where we have a board with 12 holes and 2 stores. Initially, there are 4 seeds in each hole. The game has two players who take turns to play. A turn for a player, consists of choosing one of the holes on their side of the board, picking all the seeds and then placing them in subsequent holes, one by one in an anti-clockwise direction.

```
          παίκτης A
|---|---|---|---|---|---|---|---|
| A | 4 | 4 | 4 | 4 | 4 | 4 | B |
| 0 |---|---|---|---|---|---| 0 |
|   | 4 | 4 | 4 | 4 | 4 | 4 |   |
|---|---|---|---|---|---|---|---|
          παίκτης B
```

2.1.1. Rules

- A turn for a player consists of choosing one of the holes on their side of the board, picking all the seeds and then placing them in subsequent holes, one by one in an anti-clockwise direction.
- Players store is included in sowing, but opponents store is not.
- If the last grain ends up in the players store, the player gets to play again.
- If the last grain ends up in an empty hole on the players side, the player takes all the grains in the opposite hole on the opponents side, and the players last grain, and puts it in their store. The turn ends.
- If the last grain is put anywhere else, the turn ends.
- The game ends when one player has no grains on their side, at which point all the grains on the other side are put in the store corresponding to that side.
- The winner is the player with the most grains in their store at the end of the game.

2.1.2. Demonstration

1. A starts

```
          παίκτης A
|---|---|---|---|---|---|---|---|
| A | 5 | 5 | 5 | 0 | 4 | 4 | B |
| 1 |---|---|---|---|---|---| 0 |
|   | 4 | 4 | 4 | 4 | 4 | 4 |   |
|---|---|---|---|---|---|---|---|
          παίκτης B
```

1. A continues

```
          παίκτης A
|---|---|---|---|---|---|---|---|
| A | 6 | 6 | 0 | 0 | 4 | 4 | B |
| 2 |---|---|---|---|---|---| 0 |
|   | 5 | 5 | 4 | 4 | 4 | 4 |   |
|---|---|---|---|---|---|---|---|
          παίκτης B
```

1. B plays

```
          παίκτης A
|---|---|---|---|---|---|---|---|
| A | 6 | 6 | 0 | 0 | 4 | 4 | B |
| 2 |---|---|---|---|---|---| 1 |
|   | 5 | 0 | 5 | 5 | 5 | 5 |   |
|---|---|---|---|---|---|---|---|
          παίκτης B
```

1. B plays again

```
          παίκτης A
|---|---|---|---|---|---|---|---|
| A | 6 | 6 | 0 | 0 | 4 | 4 | B |
| 2 |---|---|---|---|---|---| 1 |
|   | 0 | 1 | 6 | 6 | 6 | 6 |   |
|---|---|---|---|---|---|---|---|
          παίκτης B
```

1. A plays

```
          παίκτης A
|---|---|---|---|---|---|---|---|
| A | 7 | 7 | 1 | 1 | 0 | 4 | B |
| 2 |---|---|---|---|---|---| 1 |
|   | 0 | 1 | 6 | 6 | 6 | 6 |   |
|---|---|---|---|---|---|---|---|
          παίκτης B
```

1. B plays

```
          παίκτης A
|---|---|---|---|---|---|---|---|
| A | 7 | 7 | 1 | 1 | 1 | 5 | B |
| 2 |---|---|---|---|---|---| 2 |
|   | 0 | 1 | 0 | 7 | 7 | 7 |   |
|---|---|---|---|---|---|---|---|
          παίκτης B
```

1. A plays

```
          παίκτης A
|---|---|---|---|---|---|---|---|
| A | 8 | 8 | 2 | 2 | 2 | 0 | B |
| 2 |---|---|---|---|---|---| 1 |
|   | 0 | 1 | 6 | 6 | 6 | 6 |   |
|---|---|---|---|---|---|---|---|
          παίκτης B
```

1. B plays

```
          παίκτης A
|---|---|---|---|---|---|---|---|
| A | 8 | 8 | 2 | 2 | 2 | 0 | B |
| 2 |---|---|---|---|---|---| 1 |
|   | 0 | 0 | 7 | 6 | 6 | 6 |   |
|---|---|---|---|---|---|---|---|
          παίκτης B
```

1. A plays and captures

```
          παίκτης A
|---|---|---|---|---|---|---|---|
| A | 0 | 8 | 2 | 2 | 2 | 0 | B |
|11 |---|---|---|---|---|---| 1 |
|   | 1 | 1 | 8 | 7 | 7 | 0 |   |
|---|---|---|---|---|---|---|---|
          παίκτης B
```

1. The game continues…

3. Setting up a Development Environment

3.1. Direct Download and Installation

Qt open source installer Qt development repositories

3.2. Fedora Linux

You can install development dependencies using

```
$ sudo dnf -y groupinstall "Development Tools"
$ sudo dnf install qt6-qtbase-devel qt6-qtbase-gui qt6-qttools qt6-linguist
```

3.3. Ubuntu Linux

You can install development dependencies using

```
$ sudo apt-get install build-essential
$ sudo apt-get install qt6-base-dev qt6-base-dev-tools qt6-l10n-tools qt6-
translations-l10n qt6-tools-dev-tools
```

3.4. Connecting to a Remote Machine

You will be able to do many of the exercises on a remote machine, though you are encouraged to setup a local development environment for further work.

3.5. Linux and MacOS

Start a terminal session, then

```
$ ssh username@ip.address
```

3.6. Windows

Get Git for Windows, install it. See an installation tutorial is available from Software Carpentry.

Start a terminal session, then

```
$ ssh username@ip.address
```

3.7. Android

Install Termux and then install the OpenSSH client.

Start a terminal session, then

```
$ ssh username@ip.address
```

4. Hello World

This first section will demonstrate creating a simple hello world application. The application will be translated.

Create a folder for the tutorial, and then within that create a folder called HelloWorld

```
$ mkdir QtIntro
$ cd QtIntro
$ mkdir HelloWorld
$ cd HelloWorld
```

Now create a file called `helloworld.cpp` and enter the following content

```
#include <QApplication>
#include <QPushButton>

int main(int argc, char *argv[]) {

        QApplication app(argc, argv);
        QPushButton hello("Hello World!");
        hello.resize(250, 150);
        hello.show();

        return app.exec();
}
```

To compile the program, run

```
$ qmake -project
```

This will generate a file `HelloWorld.pro`. This is a setup file that is used to generate instructions to compile the project. The contents should be similar to

```
#############################################################################
# Automatically generated by qmake (3.1) Thu Jul 20 07:01:46 2023
#############################################################################

TEMPLATE = app
TARGET = HelloWorld
INCLUDEPATH += .

# You can make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# Please consult the documentation of the deprecated API in order to know
# how to port your code away from it.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_UP_TO=0x060000 # disables all APIs deprecated in Qt
6.0.0 and earlier

# Input
SOURCES += helloworld.cpp
```

Try compiling the file by typing

```
$ qmake
$ make
```

Compilation will fail due to missing classes that need to be included. Add two lines to the end of the file, `QT += core` and `QT += widgets` which add the missing classes that are needed to compile it.

```
#############################################################################
# Automatically generated by qmake (3.1) Thu Jul 20 07:01:46 2023
#############################################################################

TEMPLATE = app
TARGET = HelloWorld
INCLUDEPATH += .

# You can make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# Please consult the documentation of the deprecated API in order to know
# how to port your code away from it.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_UP_TO=0x060000 # disables all APIs deprecated in Qt
6.0.0 and earlier

# Input
SOURCES += helloworld.cpp
QT += core
QT += widgets
```

Now compile and run it

```
$ qmake
$ make
$ ./helloworld
```

The program runs now. You can now add a translation, first update the source file

```
// Copyright (C) 2016 The Qt Company Ltd.
// SPDX-License-Identifier: LicenseRef-Qt-Commercial OR BSD-3-Clause
// https://doc.qt.io/qt-6/qtlinguist-hellotr-example.html
//
https://code.qt.io/cgit/qt/qttools.git/tree/examples/linguist/hellotr/main.cpp?h=6.5
#include <QApplication>
#include <QPushButton>
#include <QTranslator>

int main(int argc, char *argv[]) {

    QApplication app(argc, argv);
    QTranslator translator;
    Q_UNUSED(translator.load("hellotr_gr"));
    app.installTranslator(&translator);
    QPushButton hello(QPushButton::tr("Hi!"));
    hello.resize(250, 150);
    hello.show();

    return app.exec();
}
```

Try to compile and run this

```
$ qmake
$ make
$ ./helloworld
```

Still no translations! Need to add these. There is a translation application Qt Linguist, but will just use the command line here. Create a file `hellotr_sw.ts` with the following content

```
<!DOCTYPE TS><TS>
<context>
  <name>QPushButton</name>
  <message>
    <source>Hello world!</source>
    <translation>Γεια σου κ□σμε!</translation>
  </message>
  <message>
    <source>Hi!</source>
    <translation>Γεια σου!</translation>
  </message>
</context>
</TS>
```

The translation file is not used directly, but is first compiled. On Fedora, the command is

```
$ lrelease-qt6 hellotr_sw.ts
```

Though you may need to use

```
$ lrelease hellotr_sw.ts
```

This will generate a binary file `hellotr_sw.qm`.

Update the project file to manually add the translation

```
#############################################################################
# Automatically generated by qmake (3.1) Thu Jul 20 07:01:46 2023
#############################################################################

TEMPLATE = app
TARGET = HelloWorld
INCLUDEPATH += .

# You can make your code fail to compile if you use deprecated APIs.
# In order to do so, uncomment the following line.
# Please consult the documentation of the deprecated API in order to know
# how to port your code away from it.
# You can also select to disable deprecated APIs only up to a certain version of Qt.
#DEFINES += QT_DISABLE_DEPRECATED_UP_TO=0x060000 # disables all APIs deprecated in Qt
6.0.0 and earlier

# Input
SOURCES += helloworld.cpp
TRANSLATION += hellotr_sw.ts
QT += core
QT += widgets
```

Then regenerate the makefile, recompile, and then run the application, it will be translated.

```
$ qmake
$ make
$ ./helloworld
```

5. Kalah Implementations

5.1. An Implementation for Two Human Players

You will create a command line game. Example ruby script

```
```

```ruby
#!/usr/bin/ruby
# Copyright 2020 - 2023, Benson Muite and the Nairuby contributors
# SPDX-License-Identifier: MIT
#
# A Ruby Kalah Implementaion
#
# Command line version for two players on the same computer

def boardtochar(board)
  boards=Array.new()
  (0..13).each do |i|
    if board[i] < 10
      boards[i] = " "+board[i].to_s
    else
      boards[i] = board[i].to_s
    end
  end
  return boards
end

def printboard(board)
  # The board is represented as a
  # one dimensional array
  #        Player A
  #  | 5 | 4 | 3 | 2 | 1 | 0 |
  # 6|---|---|---|---|---|---|13
  #  | 7 | 8 | 9 | 10| 11| 12|
  #        Player B
  boards = boardtochar(board)

  puts ("           A ")
  puts ("A | #{boards[5]} | #{boards[4]} | #{boards[3]} | #{boards[2]} | #{boards[1]}
| #{boards[0]} |")
  puts ("#{boards[6]}|----|----|----|----|----|----|#{boards[13]} ")
  puts ("  | #{boards[7]} | #{boards[8]} | #{boards[9]} | #{boards[10]} | #{boards[11
]} | #{boards[12]} |B")
  puts ("           B ")
end

def getmove(board,player)
   inputerror=1
   seeds=0
   # Get a valid input move
   begin
     puts ("Choose a hole #{player}")
     if player == "A"
       puts ("  |  5 |  4 |  3 |  2 |  1 |  0 |")
     else
       puts ("  |  0 |  1 |  2 |  3 |  4 |  5 |")
     end
     x = gets.chomp.to_i
```

```
    if ((x >= 0) and (x <=5))
      if ((player=="A") and (board[x]>0))
        inputerror=0
        seeds=board[x]
        board[x]=0
      elsif ((player=="B") and (board[7+x]>0))
        inputerror=0
        seeds=board[7+x]
        board[7+x]=0
      else
        puts("Please choose a hole with seeds")
      end
    else
      puts("Please input a number between 0 and 5")
    end
  end while inputerror==1

  # Place seeds in holes
  hole=0
  (0..(seeds-1)).each do |i|
    if player == "A"
      hole=x+i+1
      if hole==13
        hole+=1
      end
      hole=hole%14
    else
      hole=x+i+7+1
      if hole == 6
        hole+=1
      end
      hole=hole%14
    end
    board[hole]+=1
  end
  # Determine next player and check
  # if a capture has occurred
  oppositehole = 12-hole
  if ((hole!=6) and (player=="A"))
    player="B"
    if ((board[hole]==1) and (hole<=6))
      board[6]+=board[oppositehole]
      board[6]+=1
      board[oppositehole]=0
      board[hole]=0
    end
  elsif ((hole!=13) and (player=="B"))
    player="A"
    if ((board[hole]==1) and (hole>=7))
      board[13]+=board[oppositehole]
      board[13]+=1
```

```
        board[oppositehole]=0
        board[hole]=0
      end
    end
    return board, player
end

def checkendgame(board)

  result=-1
  azeros=0
  bzeros=0
  # count number of empty pockets for each player
  (0..5).each do |i|
    if board[i]==0
      azeros+=1
    end
    if board[i+7]==0
      bzeros+=1
    end
  end
  # If one player has empty pockets,
  # put remaining seeds in other players store
  # then determine if there is a winner or if
  # it is a draw
  if ((azeros == 6) or (bzeros == 6))
    if azeros == 6
      (7..12).each do |i|
        board[13]+=board[i]
        board[i]=0
      end
    else
      (0..5).each do |i|
        board[6]+=board[i]
        board[i]=0
      end
    end
    if board[7] > board [13]
      # A wins
      result = 0
    elsif board[13] > board[7]
      # B wins
      result = 1
    else
      # draw
      result = 2
    end
  end
  return board, result
end
```

```
# Main program
#
# Setup inital board
# The board is represented as a
# one dimensional array
#        Player A
#  | 5 | 4 | 3 | 2 | 1 | 0 |
# 6|---|---|---|---|---|---|13
#  | 7 | 8 | 9 | 10| 11| 12|
#        Player B
#      0 1 2 3 4 5 6 7 8 9 10 11 12 13
board=[4,4,4,4,4,4,0,4,4,4, 4, 4, 4,0]

# Variable to keep track of winner and whether game has ended
endgame=-1

# Starting player
player="A"

# Main Game Loop
move = 0
begin

  puts "Bao "
  printboard(board)
  board,player =  getmove(board,player)
  board, endgame = checkendgame(board)
  move +=1

end while endgame ==-1

# Show the final configuration
printboard(board)

if endgame == 0
  puts "The winner is A"
elsif endgame == 1
  puts "The winner is B"
else
  puts "Draw."
end
puts "Thanks for playing!"
```

Example C++ and Qt program

```
// Copyright 2023, Benson Muite and contributors
// SPDX-License-Identifier: MIT

#include <QTextStream>
#include <QString>
```

```
void printBoard(int board[]) {
    QTextStream out(stdout);

    out << "       1" << Qt::endl;
    out << "  | " ;
    for (int i=0; i<6; i++) {
        out << board[5-i] << " | ";
    }
    out << Qt::endl;
    out << board[6] << " | ";
    for (int i=0; i<6; i++) {
        out << "  | ";
    }
    out << board[13] << Qt::endl;
    out << "  | " ;
    for (int i=0; i<6; i++) {
        out << board[7+i] << " | ";
    }
    out << Qt::endl;
    out << "        2" << Qt::endl;
}

int getMove(int board[], int player) {
    QTextStream in(stdin);
    QTextStream out(stdout);

    int startHole;
    int finalHole;
    int seeds;
    bool inputError = true;
    out << "Player " << player << " choose a hole" << Qt::endl;
    if ( player == 1 ) {
        out << "   | 5 | 4 | 3 | 2 | 1 | 0 |" << Qt::endl;
    } else {
        out << "   | 0 | 1 | 2 | 3 | 4 | 5 |" << Qt::endl;
    }

    while (inputError) {
        in >> startHole;
        if ((startHole >= 0 ) & (startHole <= 5) ) {
            if ((player == 1) & (board[startHole] > 0)) {
                inputError=false;
                seeds=board[startHole];
                board[startHole]=0;
            } else if ((player == 2) & (board[startHole+7] > 0)) {
                inputError=false;
                seeds=board[startHole+7];
                board[startHole+7]=0;
            } else {
                out << "Please choose a hole with seeds"
```

```
                    << Qt::endl;
            }
        } else {
            out << "Please enter a value between 0 and 5"
                << Qt::endl;
        }
    }

    // Put seeds in holes
    int fill=0;
    for(int i = 0; i<seeds; i++) {
        if (player == 1) {
            fill = i+startHole+1;
            // skip opponents store
            if (fill == 13) fill +=1;
            fill = fill%14;
        } else {
            fill = i+startHole+7+1;
            // skip opponents store
            if (fill == 6) fill +=1;
            fill =fill%14;
        }
        board[fill]+=1;
    }
    finalHole=fill;

    // Check if a capture of seeds has occured
    // and determine who the next player is
    int oppositeHole = 12 - finalHole;
    if ((finalHole != 6) && (player == 1)) {
        player = 2;
        // check capture
        if ((board[finalHole] == 1) & (finalHole <= 6)) {
            board[6] += board[oppositeHole];
            board[6] += board[finalHole];
            board[finalHole] = 0;
            board[oppositeHole] = 0;
        }
    }else if((finalHole != 13) && (player == 2)) {
        player = 1;
        // check capture
        if ((board[finalHole] == 1) & (finalHole >= 7)) {
            board[13] += board[oppositeHole];
            board[13] += board[finalHole];
            board[finalHole] = 0;
            board[oppositeHole] = 0;
        }
    }
    return player;
}
```

```
bool checkEndGame(int board[]) {

    QTextStream out(stdout);

    bool endGame=false;
    int seedsOnSide[] = {0,0};
    // Check if a player has only empty holes
    for (int i=0; i<6; i++) seedsOnSide[0] += board[i];
    for (int i=7; i<13; i++) seedsOnSide[1] += board[i];
    if ((seedsOnSide[0] == 0) | (seedsOnSide[1] = 0)) {
        endGame = true;
        // Check the winner
        // Count total seeds in stores
        board[13] += seedsOnSide[1];
        board[6] += seedsOnSide[0];
        if (board[6] == board[13]) {
            out << "A draw" << Qt::endl;
        } else if ( board[6] > board[13] ) {
            out << "Player 1 wins" << Qt::endl;
        } else {
            out << "Player 2 wins" << Qt::endl;
        }
    }
    return endGame;
}

int main(void) {

/*
      Player 1
    | 5 | 4 | 3 | 2 | 1 | 0 |
   6|---|---|---|---|---|---|13
    | 7 | 8 | 9 | 10| 11| 12|
          Player 2
*/
    int board[14] = {4,4,4,4,4,4,0,
                     4,4,4,4,4,4,0};

    int player = 1;
    bool endGame = false;
    while (!endGame) {
        printBoard(board);
        player = getMove(board,player);
        endGame = checkEndGame(board);
    }

    return 0;
}
```

5.2. A Computer Opponent

Example in Ruby

```
#!/usr/bin/ruby
# Copyright 2020 - 2023, Benson Muite and the Nairuby contributors
# SPDX-License-Identifier: MIT
#
# A Ruby Kalah Implementaion
#
# Command line version for two players on the same computer

def boardtochar(board)
  boards=Array.new()
  (0..13).each do |i|
    if board[i] < 10
      boards[i] = " "+board[i].to_s
    else
      boards[i] = board[i].to_s
    end
  end
  return boards
end

def printboard(board)
  # The board is represented as a
  # one dimensional array
  #        Player A
  #  | 5 | 4 | 3 | 2 | 1 | 0 |
  # 6|---|---|---|---|---|---|13
  #  | 7 | 8 | 9 | 10| 11| 12|
  #        Player B
  boards = boardtochar(board)

  puts ("          A ")
  puts ("A | #{boards[5]} | #{boards[4]} | #{boards[3]} | #{boards[2]} | #{boards[1]}
| #{boards[0]} |")
  puts ("#{boards[6]}|----|----|----|----|----|----|#{boards[13]} ")
  puts ("  | #{boards[7]} | #{boards[8]} | #{boards[9]} | #{boards[10]} | #{boards[11
]} | #{boards[12]} |B")
  puts ("          B ")
end

def gethumanmove(board,player)
   inputerror=1
   seeds=0
   # Get a valid input move
   begin
     puts ("Chagua mfuko mchezaji #{player}")
     if player == "A"
```

```
      puts ("  |  5 |  4 |  3 |  2 |  1 |  0 |")
    else
      puts ("  |  0 |  1 |  2 |  3 |  4 |  5 |")
    end
    x = gets.chomp.to_i
    if ((x >= 0) and (x <=5))
      if ((player=="A") and (board[x]>0))
        inputerror=0
        seeds=board[x]
        board[x]=0
      elsif ((player=="B") and (board[7+x]>0))
        inputerror=0
        seeds=board[7+x]
        board[7+x]=0
      else
        puts("Please choose a hole with seeds")
      end
    else
      puts("Please enter a number between 0 and 5")
    end
  end while inputerror == 1
  board,player = placeseeds(board,seeds,x,player)
  return board,player
end

def getcomputermove(board,player)
  inputerror=1
  seeds=0
  # Get a valid input move
  begin
    x = ((10000*rand).floor)%6
    if ((player=="A") and (board[x]>0))
      inputerror=0
      seeds=board[x]
      board[x]=0
    elsif ((player=="B") and (board[7+x]>0))
      inputerror=0
      seeds=board[7+x]
      board[7+x]=0
    end
  end while inputerror == 1
  board,player = placeseeds(board,seeds,x,player)
  return board,player
end

def placeseeds(board,seeds,x,player)

  # Place seeds in holes
  hole=0
  (0..(seeds-1)).each do |i|
    if player == "A"
```

```
        hole=x+i+1
        if hole==13
          hole+=1
        end
        hole=hole%14
      else
        hole=x+i+7+1
        if hole == 6
          hole+=1
        end
        hole=hole%14
      end
      board[hole]+=1
    end
    # Determine next player and check
    # if a capture has occurred
    oppositehole = 12-hole
    if ((hole!=6) and (player=="A"))
      player="B"
      if ((board[hole]==1) and (hole<=6))
        board[6]+=board[oppositehole]
        board[6]+=1
        board[oppositehole]=0
        board[hole]=0
      end
    elsif ((hole!=13) and (player=="B"))
      player="A"
      if ((board[hole]==1) and (hole >=7))
        board[13]+=board[oppositehole]
        board[13]+=1
        board[oppositehole]=0
        board[hole]=0
      end
    end
    return board, player
end

def checkendgame(board)

  result=-1
  azeros=0
  bzeros=0
  # count number of empty pockets for each player
  (0..5).each do |i|
    if board[i]==0
      azeros+=1
    end
    if board[i+7]==0
      bzeros+=1
    end
  end
```

```
  # If one player has empty pockets,
  # put remaining seeds in other players store
  # then determine if there is a winner or if
  # it is a draw
  if ((azeros == 6) or (bzeros == 6))
    if azeros == 6
      (7..12).each do |i|
        board[13]+=board[i]
        board[i]=0
      end
    else
      (0..5).each do |i|
        board[6]+=board[i]
        board[i]=0
      end
    end
    if board[7] > board [13]
      # A wins
      result = 0
    elsif board[13] > board[7]
      # B wins
      result = 1
    else
      # draw
      result = 2
    end
  end
  return board, result
end

# Main program
#
# Setup inital board
# The board is represented as a
# one dimensional array
#        Player A
#  | 5 | 4 | 3 | 2 | 1 | 0 |
# 6|---|---|---|---|---|---|13
#  | 7 | 8 | 9 | 10| 11| 12|
#        Player B
#      0 1 2 3 4 5 6 7 8 9 10 11 12 13
board=[4,4,4,4,4,4,0,4,4,4, 4, 4, 4,0]

# Variable to keep track of winner and whether game has ended
endgame=-1

# Starting player
puts("Choose the starting player")
puts("If you want to start enter A")
puts("If you want the computer to start enter B")
```

```
begin
  start = gets.chomp.to_s
  if start == "A"
    human="A"
    computer="B"
  elsif start == "B"
    human="B"
    computer="A"
  else
    puts ("Please ente A or B")
  end
end while ((start!="A") and (start!="B"))

player="A"

# Main Game Loop
move = 0
begin

  puts("Bao ")
  printboard(board)
  if player == human
    board,player =  gethumanmove(board,player)
  else
    board,player = getcomputermove(board,player)
  end
  board, endgame = checkendgame(board)
  move +=1

end while endgame ==-1

# Show the final configuration
printboard(board)

if endgame == 0
  puts("A won")
elsif endgame == 1
  puts("B won")
else
  puts("Draw")
end
puts("Thanks for playing!")
```

Example Qt and C++ program

```
// Copyright 2023, Benson Muite and contributors
// SPDX-License-Identifier: MIT

#include <QTextStream>
#include <QString>
```

```
#include <QTime>
#include <QRandomGenerator>

void printBoard(int board[]) {
    QTextStream out(stdout);

    out << "       1" << Qt::endl;
    out << "  | " ;
    for (int i=0; i<6; i++) {
        out << board[5-i] << " | ";
    }
    out << Qt::endl;
    out << board[6] << " | ";
    for (int i=0; i<6; i++) {
        out << "  | ";
    }
    out << board[13] << Qt::endl;
    out << "  | " ;
    for (int i=0; i<6; i++) {
        out << board[7+i] << " | ";
    }
    out << Qt::endl;
    out << "        2" << Qt::endl;
}

int getMove(int board[], int player) {
    QTextStream in(stdin);
    QTextStream out(stdout);

    int startHole;
    bool inputError = true;
    out << "Player " << player << " choose a hole" << Qt::endl;
    if ( player == 1 ) {
        out << "   | 5 | 4 | 3 | 2 | 1 | 0 |" << Qt::endl;
    } else {
        out << "   | 0 | 1 | 2 | 3 | 4 | 5 |" << Qt::endl;
    }

    while (inputError) {
        in >> startHole;
        if ((startHole >= 0 ) & (startHole <= 5) ) {
            if ((player == 1) & (board[startHole] > 0)) {
                inputError=false;
            } else if ((player == 2) & (board[startHole+7] > 0)) {
                inputError=false;
            } else {
                out << "Please choose a hole with seeds"
                    << Qt::endl;
            }
        } else {
            out << "Please enter a value between 0 and 5"
```

```
                << Qt::endl;
        }
    }

    return startHole;
}

int updateBoard(int board[], int player, int startHole) {

        int seeds;
    int fill=0;
    int finalHole;
        if (player == 1) {
        seeds = board[startHole];
        board[startHole] = 0;
    } else {
        seeds = board[startHole+7];
        board[startHole+7] = 0;
    }

    // Put seeds in holes
    for(int i = 0; i<seeds; i++) {
        if (player == 1) {
            fill = i+startHole+1;
            // skip opponents store
            if (fill == 13) fill += 1;
            fill = fill%14;
        } else {
            fill = i + startHole + 7 + 1;
            // skip opponents store
            if (fill == 6) fill += 1;
            fill = fill%14;
        }
        board[fill] += 1;
    }
    finalHole = fill;

    // Check if a capture of seeds has occured
    // and determine who the next player is
    int oppositeHole = 12 - finalHole;
    if ((finalHole != 6) && (player == 1)) {
        player = 2;
        // check capture
        if ((board[finalHole] == 1) & (finalHole <= 6)) {
            board[6] += board[oppositeHole];
            board[6] += board[finalHole];
            board[finalHole] = 0;
            board[oppositeHole] = 0;
        }
    }else if((finalHole != 13) && (player == 2)) {
        player = 1;
```

```
        // check capture
        if ((board[finalHole] == 1) & (finalHole >= 7)) {
            board[13] += board[oppositeHole];
            board[13] += board[finalHole];
            board[finalHole] = 0;
            board[oppositeHole] = 0;
        }
    }
    return player;
}

int getComputerMove(int board[], int player) {

    bool inputError=true;
    int startHole;
        QTime time = QTime::currentTime();
        QRandomGenerator rng((uint) time.msec());

    while (inputError) {
        startHole = rng.generate() % 6;
        if ((player == 1) & (board[startHole] > 0)) {
            inputError = false;
        } else if ((player == 2) & (board[startHole+7] > 0)) {
            inputError = false;
        }
    }

    return startHole;
}

bool checkEndGame(int board[]) {

    QTextStream out(stdout);

    bool endGame = false;
    int seedsOnSide[] = {0,0};
    // Check if a player has only empty holes
    for (int i = 0; i < 6; i++) seedsOnSide[0] += board[i];
    for (int i = 7; i < 13; i++) seedsOnSide[1] += board[i];
    if ((seedsOnSide[0] == 0) | (seedsOnSide[1] == 0)) {
        endGame = true;
        // Check the winner
        // Count total seeds in stores
        board[13] += seedsOnSide[1];
        board[6] += seedsOnSide[0];
        if (board[6] == board[13]) {
            out << "A draw" << Qt::endl;
        } else if ( board[6] > board[13] ) {
            out << "Player 1 wins" << Qt::endl;
        } else {
            out << "Player 2 wins" << Qt::endl;
```

```
        }
    }
    return endGame;
}

int chooseStartingPlayer() {
    QTextStream in(stdin);
    QTextStream out(stdout);

    int player = 0;
    bool repeat = true;
    while (repeat) {
        out << "Enter 1 for you to start or 2 for the computer to start"
            << Qt::endl;
        in >> player;
        if ((player == 1) | (player == 2)) repeat = false;
    }

    return player;
}

int main(void) {

/*
      Player 1
    | 5 | 4 | 3 | 2 | 1 | 0 |
   6|---|---|---|---|---|---|13
    | 7 | 8 | 9 | 10| 11| 12|
          Player 2
*/
    int board[14] = {4,4,4,4,4,4,0,
                     4,4,4,4,4,4,0};

    int player = 1;
    bool endGame = false;
    int startHole = 0;
    player = chooseStartingPlayer();
    while (!endGame) {
        printBoard(board);
        if (player == 1) {
            startHole = getMove(board,player);
        } else {
            startHole = getComputerMove(board,player);
        }
        player = updateBoard(board,player,startHole);
        endGame = checkEndGame(board);
    }

    return 0;
}
```

5.3. Translated Game with Computer Opponent

```
// Copyright 2023, Benson Muite and contributors
// SPDX-License-Identifier: MIT

#include <QTextStream>
#include <QString>
#include <QTime>
#include <QRandomGenerator>
#include <QTranslator>
#include <QCoreApplication>

void printBoard(int board[]) {
    QTextStream out(stdout);

    out << "       1" << Qt::endl;
    out << "  | " ;
    for (int i=0; i<6; i++) {
        out << board[5-i] << " | ";
    }
    out << Qt::endl;
    out << board[6] << " | ";
    for (int i=0; i<6; i++) {
        out << "  | ";
    }
    out << board[13] << Qt::endl;
    out << "  | " ;
    for (int i=0; i<6; i++) {
        out << board[7+i] << " | ";
    }
    out << Qt::endl;
    out << "        2" << Qt::endl;
}

int getMove(int board[], int player) {

    QTextStream in(stdin);
    QTextStream out(stdout);

    int startHole;
    bool inputError = true;
    out << QCoreApplication::translate("get1","Player ") << player
        << QCoreApplication::translate("get2"," choose a hole") << Qt::endl;
    if ( player == 1 ) {
        out << "   | 5 | 4 | 3 | 2 | 1 | 0 |" << Qt::endl;
    } else {
        out << "   | 0 | 1 | 2 | 3 | 4 | 5 |" << Qt::endl;
    }

    while (inputError) {
```

```
        in >> startHole;
        if ((startHole >= 0 ) & (startHole <= 5) ) {
            if ((player == 1) & (board[startHole] > 0)) {
                inputError=false;
            } else if ((player == 2) & (board[startHole+7] > 0)) {
                inputError=false;
            } else {
                out << QCoreApplication::translate("get3","Please choose a hole with
seeds")
                    << Qt::endl;
            }
        } else {
            out << QCoreApplication::translate("get4","Please enter a value between 0
and 5")
                << Qt::endl;
        }
    }

    return startHole;
}

int updateBoard(int board[], int player, int startHole) {

        int seeds;
    int fill = 0;
    int finalHole;
        if (player == 1) {
        seeds = board[startHole];
        board[startHole] = 0;
    } else {
        seeds = board[startHole+7];
        board[startHole+7] = 0;
    }

    // Put seeds in holes
    for(int i = 0; i<seeds; i++) {
        if (player == 1) {
            fill = i+startHole+1;
            // skip opponents store
            if (fill == 13) fill += 1;
            fill = fill%14;
        } else {
            fill = i + startHole + 7 + 1;
            // skip opponents store
            if (fill == 6) fill += 1;
            fill = fill%14;
        }
        board[fill] += 1;
    }
    finalHole = fill;
```

```
    // Check if a capture of seeds has occured
    // and determine who the next player is
    int oppositeHole = 12 - finalHole;
    if ((finalHole != 6) && (player == 1)) {
        player = 2;
        // check capture
        if ((board[finalHole] == 1) & (finalHole <= 6)) {
            board[6] += board[oppositeHole];
            board[6] += board[finalHole];
            board[finalHole] = 0;
            board[oppositeHole] = 0;
        }
    } else if((finalHole != 13) && (player == 2)) {
        player = 1;
        // check capture
        if ((board[finalHole] == 1) & (finalHole >= 7)) {
            board[13] += board[oppositeHole];
            board[13] += board[finalHole];
            board[finalHole] = 0;
            board[oppositeHole] = 0;
        }
    }
    return player;
}

int getComputerMove(int board[], int player) {

    bool inputError=true;
    int startHole;
        QTime time = QTime::currentTime();
        QRandomGenerator rng((uint) time.msec());

    while (inputError) {
        startHole = rng.generate() % 6;
        if ((player == 1) & (board[startHole] > 0)) {
            inputError = false;
        } else if ((player == 2) & (board[startHole+7] > 0)) {
            inputError = false;
        }
    }

    return startHole;
}

bool checkEndGame(int board[]) {
    QTextStream out(stdout);

    bool endGame = false;
    int seedsOnSide[] = {0,0};
    // Check if a player has only empty holes
    for (int i = 0; i < 6; i++) seedsOnSide[0] += board[i];
```

```
    for (int i = 7; i < 13; i++) seedsOnSide[1] += board[i];
    if ((seedsOnSide[0] == 0) | (seedsOnSide[1] == 0)) {
        endGame = true;
        // Check the winner
        // Count total seeds in stores
        board[13] += seedsOnSide[1];
        board[6] += seedsOnSide[0];
        if (board[6] == board[13]) {
            out << QCoreApplication::translate("end1","A draw") << Qt::endl;
        } else if ( board[6] > board[13] ) {
            out << QCoreApplication::translate("end2","Player 1 wins") << Qt::endl;
        } else {
            out << QCoreApplication::translate("end3","Player 2 wins") << Qt::endl;
        }
    }
    return endGame;
}

int chooseStartingPlayer() {
    QTextStream in(stdin);
    QTextStream out(stdout);

    int player = 0;
    bool repeat = true;
    while (repeat) {
        out << QCoreApplication::translate("start1","Enter 1 for you to start or 2 for
the computer to start")
            << Qt::endl;
        in >> player;
        if ((player == 1) | (player == 2)) repeat = false;
    }

    return player;
}

int main(int argc, char *argv[]) {

    QCoreApplication app(argc, argv);
    // Setup translation files
    QTranslator translator;
    Q_UNUSED(translator.load("kalahtr_sw"));
    QCoreApplication::installTranslator(&translator);

    /*
                Player 1
         | 5 | 4 | 3 | 2 | 1 | 0 |
        6|---|---|---|---|---|---|13
         | 7 | 8 | 9 | 10| 11| 12|
                Player 2
         */
    int board[14] = {4,4,4,4,4,4,0,
```

```
                4,4,4,4,4,4,0};

    int player = 1;
    bool endGame = false;
    int startHole = 0;

    player = chooseStartingPlayer();
        // Main game loop
    while (!endGame) {
        printBoard(board);
        if (player == 1) {
            startHole = getMove(board,player);
        } else {
            startHole = getComputerMove(board,player);
        }
        player = updateBoard(board,player,startHole);
        endGame = checkEndGame(board);
    }

    return 0;
}
```

The corresponding translation file is

```
<!DOCTYPE TS><TS>
<context>
  <name>get1</name>
  <message>
    <source>Player </source>
    <translation>Mchezaji</translation>
  </message>
  <name>get2</name>
  <message>
    <source> choose a hole</source>
    <translation> chagua shimo</translation>
  </message>
  <name>get3</name>
  <message>
    <source>Please choose a hole with seeds</source>
    <translation>Tafadhali chagua shimo ina mbegu</translation>
  </message>
  <name>get4</name>
  <message>
    <source>Please enter a value between 0 and 5</source>
    <translation>Tafadhali ingiza nambari katika 0 na 5</translation>
  </message>
  <name>end1</name>
  <message>
    <source>A draw</source>
    <translation>Mchezo ni sare</translation>
  </message>
  <name>end2</name>
  <message>
    <source>Player 1 wins</source>
    <translation>Mshindi ni mchezaji 1</translation>
  </message>
  <name>end3</name>
  <message>
    <source>Player 2 wins</source>
    <translation>Mshindi ni mchezaji 2</translation>
  </message>
  <name>start1</name>
  <message>
    <source>Enter 1 for you to start or 2 for the computer to start</source>
    <translation>Ingiza 1 kuanza au 2 kichakato ianze</translation>
  </message>
</context>
</TS>
```

Be sure to add the name of the translation file to the project file, then generate or regenerate the Makefile, compile the source code and run your program.

6. Next Steps

- Robust input validation
- Improved text output alignment when there are more than 9 seeds in a hole
- Graphical user interface
- More sophisticated computer opponents
- Awale implementation

7. References

- Geoffrey Irving, Jeroen Donkers and Jos Uiterwijk, Solving Kalah
- Qt for Beginners
- Qt 6.5 Documentation
- Getting Started with Qt on the Commandline
- Jan Bodnar, ZetCode Qt5 tutorial
- Wikipedia, Mancala
- Alain Le Bot, Laurent Le Bot, Diana Martin de Argenta and Christian Gruber, Free Awale
- Wikipedia, Awale
- Nairuby Games, Bao
- Harsh Kumar, Oware