Bubble Kid QML game

Introduction

05 Dec 2010

Bubble Kid is a very simple action game written for children with two to four years old. The game challenge is to pop all bubbles that appears on the screen as quickly as possible. It has three speed levels and three different number of bubbles that can be combined to generate several difficult levels. Bubble Kid was written in QML (Qt 4.7) and it was inspired from Same Game.

Game components

The game consists in a background image (for personal purposes only) over which the bubbles are animated, as you can see below:

A periodic timer is used to trigger the animation routine **onUpdate()**. This timer will control the game speed and updating the game scene more frequently when a faster game is selected. At the bottom, a small toolbar is placed for game control and show the score. Four dialogs complete the game scene: number of bubbles, bubbles' speed, game result and about. All these elements are created in the file bubblekids.qml.

The game logic is done using a javascript file named game.js. When the user presses **Start**, the function **startNewGame()** is called and bubbles are dynamically created according to the current speed and number of bubbles. The original position and direction are randomly generated by **createBlock()** function, a modified version of original createBlock() presented in Same Game.

```
function createBlock(n)
{
    if (component == null)
        component = Qt.createComponent("Bubble.qml");

    if (component.status == Component.Ready)
    {
        var obj = component.createObject(background);
        if (obj == null) {
            console.log("error creating bubble");
            console.log(component.errorString());
            return false;
        }
        obj.index = n;
        obj.width = blockSize;
        obj.height = blockSize;
        obj.x = Math.random()*(background.width-blockSize);
        obj.y = Math.random()*(background.height-blockSize);
        obj.dx = Math.random() > 0.5 ? bubbleStep : -bubbleStep;
        obj.dy = Math.random() > 0.5 ? bubbleStep : -bubbleStep;
        board[n] = obj;
    }
    else
    {
        console.log("error loading bubble component");
        console.log(component.errorString());
        return false;
    }
    return true;
}
```

The timer is enabled to update the game scene at regular time intervals related to the current speed level.

```
Timer {
    id: gameTimer
    interval: Game.getUpdateTime()
    repeat: true
    running: false
```


(C) Copyright Nokia 2013. All rights reserved.

```
    triggeredOnStart: false
    onTriggered: Game.onUpdate()
}
```

A MouseArea element is used to detect clicks and verify when a bubble popped.

```
    Item {
        width: parent.width
        anchors { top: parent.top; bottom: toolBar.top }

        Image {
            id: background
            anchors.fill: parent
            source: "bubblekid-640x360.jpg"
            fillMode: Image.PreserveAspectCrop

            MouseArea {
                id: buttonMouseArea
                anchors.fill: parent
                acceptedButtons: Qt.LeftButton
                onClicked: {
                    Game.removeBubble(mouseX,mouseY)
                }
            }
        }
    }
```

This check is performed by the function **removeBubble()** that verifies if the mouse click was really done over an existing bubble or not. This function is responsible to check when the game overs evaluating the number of remaining bubble in the game scene as well.

```
function removeBubble(mx,my)
{
    if(!gameStarted)
        return;

    for (var n = 0; n < numBubbles; n++)
    {
        if(board[n].visible)
        {
            if( (mx > board[n].x) && (mx < (board[n].x + board[n].width)) &&
                (my > board[n].y) && (my < (board[n].y + board[n].height)) )
            {
                board[n].visible = false;
                ++bubblesClicked;
                break;
            }
        }
    }
    if(bubblesClicked >= numBubbles)
    {
        gameOverDialog.show("Congratulations !\n" +
                            "You popped all bubbles in\n" +
                            score + " seconds !");
        newGameBut.text = " Start "
        gameStarted = false;
        bubblesClicked = 0;
        gameTimer.stop();
    }
}
```

Game setup

Two dialogs are used to set the difficult level in the game. Each dialog is created as a text message and some buttons with the options inside. For instance, below is presented the speed level dialog and the corresponding QML code (file SpeedDlg.qml):

```
import Qt 4.7

Rectangle {
    id: container
    smooth: true
    radius: 8
    color: "khaki"
```


(C) Copyright Nokia 2013. All rights reserved.

```
    border.color: "black"
    border.width: 2
    property int selected: -1
    signal speedSelected

    width: dialogText.width
    height: dialogText.height + turtleSpeed.height +
            rabbitSpeed.height + guepardSpeed.height + 10 + 3*5
    opacity: 0

    function show(text) {
        container.opacity = 1;
    }

    function hide() {
        container.opacity = 0;
    }

    MouseArea {
        anchors.fill: parent
        onClicked: hide();
    }

    Column {
        id: optionsColumn1
        spacing: 10

        Text {
            id: dialogText
            font.pointSize: 12
            text: "  How fast are you?  "
        }

        Column {
            id: optionsColumn
            anchors.horizontalCenter: parent.horizontalCenter
            spacing: 5

            Button {
                id: turtleSpeed
                text: "Turtle"
                textPointSize: 10
                width: dialogText.width - 10
                radius: 3
                onClicked: {
                    selected = 0;
                    speedSelected();
                }

            }
            Button {
                id: rabbitSpeed
                text: "Rabbit"
                textPointSize: 10
                width: dialogText.width - 10
                radius: 3
                onClicked: {
                    selected = 1;
                    speedSelected();
                }
            }

            Button {
                id: guepardSpeed
                text: "Guepard"
                textPointSize: 10
                width: dialogText.width - 10
                radius: 3
                onClicked: {
                    selected = 2;
                    speedSelected();
                }
            }
        }
    }
}
```

When the user clicks over an option, a custom signal is generated (**speedSelected**) and handled by the dialog element in **bubblekid.qml**. Again, the game logic is called to setup the proper level (in this case, the function **setGameSpeed()**). A custom attribute is used to hold the current setup (**speedDialog.selected**):

```
SpeedDlg {
    id: speedDialog
    anchors.centerIn: parent
    z: 100
    onSpeedSelected: {
        screen.state = "hideSpeedDialog"
        Game.setGameSpeed(speedDialog.selected);
        speedDialog.hide();
    }
}
```

Setup dialogs (speed or number of bubbles) are animated using transitions based on opacity property. The idea is to show gradually the dialog while the toolbar disappears:

```
    states: [
        State {
            name: "showSpeedDialog";
            PropertyChanges { target: toolBar; opacity:0}
            PropertyChanges { target: speedDialog; opacity:1}
        },
        State {
            name: "hideSpeedDialog";
            PropertyChanges { target: toolBar; opacity:1}
            PropertyChanges { target: speedDialog; opacity:0}
        },
        State {
            name: "showBubbleDialog";
            PropertyChanges { target: toolBar; opacity:0}
            PropertyChanges { target: bubbleDialog; opacity:1}
        },
```


(C) Copyright Nokia 2013. All rights reserved.

```
        State {
            name: "hideBubbleDialog";
            PropertyChanges { target: toolBar; opacity:1}
            PropertyChanges { target: bubbleDialog; opacity:0}
        }

    ]
    transitions: [
        Transition {
            to: "*"
            NumberAnimation {
                properties: "opacity"
                duration: 500
                easing.type: Easing.InOutQuad
            }
        }
    ]
```

Wrap up

For generating the game executable (.exe or .sis), all qml files are saved into a resource file and hence automatically added to the final executable. Files can be read after using the prefix **"qrc:/"**. Moreover, a QML View is created using the object QDeclarativeView as main window (see file bubblekid.cpp):

```
int main(int argc, char *argv[])
{
    QApplication application(argc, argv);
    QDeclarativeView view;
    view.setSource(QUrl("qrc:/bubblekid.qml"));
    view.setResizeMode(QDeclarativeView::SizeRootObjectToView);
    QObject::connect(view.engine(), SIGNAL(quit()), &application, SLOT(quit()));

#if defined(Q_OS_SYMBIAN)
    view.showFullScreen();
#else
    view.setGeometry(QRect(100, 100, 640, 360));
    view.show();
#endif
    return application.exec();
}
```

Future enhancements

For a better game experience in Bubble Kid, several enhancements are possible:

- Sound (using Qt Mobility) and animation when a bubble is popped.
- Persistent scores, showing the best players.
- More and better transitions.
- More randomness when animating the bubbles.
- Buttons with images on it, helping children to select the proper option.

Source code

The source code is available at Nokia Project Repository as a public repository. If you want to take part in this project, just comment the article or send me a message.

Video

Below, you can watch Bubble Kid in action:

The media player is loading...

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.