

ICS3C
 INTRODUCTION TO COMPUTER PROGRAMMING/COMPUTER SCIENCE
 COURSE OUTLINE AND EVALUATION

1. **TOPICS**

- a) Introduction to C#
- b) Designing a User Interface
- c) The Basics of writing code
- d) Entering and outputting data
- e) Selective processing
- f) Repetitive processing
- g) Problem solving techniques
- h) Methods
- i) Simple file concepts
- j) String method applications
- k) One dimensional arrays
- l) Probabilistic simulations
- m) Graphics techniques
 - 1. Intro concepts
 - 2. Mouse Events
 - 3. Animation techniques
 - 4. Keyboard events
 - 5. Sound/Video
- n) **Robotics** using LegoMindstorms

2. **EVALUATION**

Ongoing Assessment and Evaluation 70%			
Knowledge and Understanding 30%	Thinking and Inquiry 15%	Communication 10%	Application 15%
- unit tests - quizzes	- lab work - projects - independent study	- notebook - journal/blog - presentations	- assignments - projects - lab work
Final Evaluation 30%			
- Formal Exam: Written and Application (15%) - Final Course Project (15%)			

ICS3C (2009-2010)

INTRODUCTION TO COMPUTER PROGRAMMING/COMPUTER SCIENCE

1. INTRODUCTION

First day of classes (1/2 period unit)

- logging in and traversing the menu structure
- overview of using FIRSTCLASS
- demo of sjbdownload site/blogs

2. PROGRAMMING AND PROBLEM SOLVING USING C#

a) Introduction to C# - Using Visual Studio Express Edition

- intro to environment
- using *sample* illustrate
 - opening/running/viewing code (design mode/ code mode)
 - solution explorer window/design window/toolbox

b) Intro to C# coding

- Console App – ch02_Examples (*ch02_Helloworld/ch02_ProgrammingMessage*)
 - includes ReadLine()/
 - discuss Console.WriteLine()/Console.Write();
- Windows App – ch02 (*ASimpleProgram*)
 - includes labels/picturebox (no regular image control like vb)

c) Creating Your First Apps – Designing a User Interface

- (1) Labels, Textboxes, Buttons and PictureBox (use local resources)
 - coding simple events (use Save All)
 - chapter1 – *VisualHelloWorld*
 - note: F7 opens code window
 - include Console.Beep()/MessageBox.Show()/Exit Button -> this.close(); or Application.Exit
 - attaching different events to controls (click on lightning bolt) eg. MouseOver

=> Videos (01-csharp/02-csharp/03-csharp) + See Intro Notes (cshtp2_02.pdf/csharpkidspart1)

(2) More User Interface and Events

- suggested C# prefixes for naming objects (**see sheet**)
- more buttons (*FormFun*) – note form width/height/backcolor/demo setting background image prop
- radionbuttons
 - using a groupbox (use hellod example from VB)
 - note: need to use Form1_Activated -> then initialize Rdo to checked false and Pic to visible false
 - adding images to a picturebox (local vs proj resource – stored in resource dir)
- A detailed look at the Explorer Window in a windows app
 - Form1.cs -> Form1.Designer.cs **** Remember to **Rename** to *Frmprojectanme* ****
 - may need to modify contents when debuggin due to adding rogue control and code
 - Program.cs (contains main())
- Take a look at folder contents of a solution

=> Problem (Radio Button Problem) + csharp_04 Creating a user interface/csharp_05 Handling events/csharp_06 setting properties

(3) Scroll Bars and NumericUpDown

- scrollbars (note: use of Convert.ToString ())
- *rgbColors* (numeric up down) -> note: use of casting (int)nudRed.value

(4) Menus

- ch14 (*menus*)
- then do simple example (colors/size) -> this.WindowState=FormWindowState.Maximized

=> Problem

(5) Working with multiple forms (A brief intro – See *MultiForm*)

- adding new forms / right click project – add new item – windows form
- .show/.close()
 - FrmTwo Two = new FrmTwo();
 - Two.Show();
 - this.Close();
- do example similar to sjb café (intro/main)
- set opacity property of form (semi-transparent)
- change the startup form
 - need to modify Program.cs -> Application.Run(new FrmOne())
 - then demo *BeforeAndAfterStartup*

=> Problem

=>QUIZ (short answer + practical)

d) **Entering and Outputting Data**

(1) The Concept of Computer Storage

- variables/data types (int/double/string)
- declaring a constant -> const double PI=3.14
- assignment statements (we do problem1/2)
- using comments
- simple program -> int counter=3;
 MessageBox.Show(counter.ToString());
- **scope of variables (local/public)**

=> videos (04csharp/csharp_07)

(2) Mathematical Applications

- arithmetic operators (Math.pow(x,2)/ % modulus/integer arithmetic (17/3)
- casting (double) 5/2 vs (double) (5/2)
- Examples (Win Apps)
 - *savings* (note use of Convert.ToInt32)
 - *average* (note use of vb style inputbox -> add reference to Microsoft.VisualBasic)
 - also note use of \n \t and casting due to integer arithmetic
 - may not work on network at school -> run from c:
- Examples (Console)
 - *simpleaddition*
 - *carpetexample*
 - *ageincrementer* (note use of int.Parse()/double.Parse())
 - *squareinput*
- Simple quotient – remainder example (recall feet and inches problem from grade 10)

=> Mathematical Applications Problems (1-7) (Do first one with students)

WORK PERIOD

=> Take up problems

- Sample Application (*pricequote*) –Venus Motor Sales
 - converting numbers to formatted strings
 - x.ToString("c") currency
 - x.ToString("n2") two decimal places
 - tab order (view)

=> Case Problems 1

WORK PERIOD

TEST (theoretical/practical)

e) **Selection-Repetition**

(1) Selective Processing

- Intro**
- *GuessNumber* (also note: != doesn't equal / == /&& ||)
- redo ticket example from grade 10
- *SeeSharpQuiz*
- *ProvincePicker* (switch example)
- Math applications (determine whether a number is even or odd)

=> Selective Processing Problems #1-3

- selective processing problem #4 (ccf)
- another example (*InvoiceTotal*)

=> Selective Processing Problems #5-6 (use problem #4 as basis for solving new problems - #7 is optional)

Sample Applications

- includes use of checkboxes and radiobuttons
 - *sandwich*
 - *tiictactoe* (note use of textbox for black lines)
 - note click event with offscreen btnMark.PerformClick()
-

(2) User Defined Dialogs

- **SimpleDialog**
- **messageboxes** (DialogResult r = MessageBox.Show() / if (r==DialogResult.No)
- **OpeningandClosingWindows**
- * **CustomDialogs**
 - Show vs ShowDialog
 - if (nameDialog.ShowDialog() == DialogResult.OK) vs
 - DialogResult r = nameDialog.ShowDialog();
 - If (r==DialogResult.OK)
 - using class properties
 - get {return TxtBox.Text;} in public string UserName
 - setting DialogResult to OK/Cancel for button
 - also set Form properties (AcceptButton and CancelButton)
- Monitoring keystrokes using the Keypress event
 - **KeypressDemo**

=> Problem: MainForm/Patient Info Form + Csharp_09 Program Flow (preview/review)
(PatientInfo)

WORK PERIOD

(3) Repetitive Processing

Counter controlled repetition (while)

- **counter** (experiment with different combinations – decrement/increment --/++/+=/-=)
- **accumulator (these are console apps - add Console.ReadLine() to last line so display remains on screen)**

User Controlled repetition (while/do while)

- **UserControlledRepetition** (uses inputbox run from c:)

For loops

- **for** (experiment with different combinations)
- **nestedloop**

Sample applications (RepApps) (uses inputbox run from c⊙)

- area of circle (note Math.PI)
- **conversion** (while) -> $f=c*9/5+32$ (note use of Environment.NewLine instead of \n)
- **sum**(for)
- **combo** (for)

=> Problems

WORK PERIOD

=> take up problems + Review/Extension (**EvenOdd**)

User controlled repetition Revisited (with custom dialog)

- note: (int) e.KeyChar==8

=> Selection-Repetition Assignment (Due in 3 periods) -> See AgeHeight.exe

WORK PERIOD

WORK PERIOD

WORK PERIOD

Combo Boxes and List Boxes

- **ComboBoxDemo** (Items property – collections) + SelectedIndex property
- Demo use of List box to display large lists
- **ListBoxExample**

REVIEW/WORK PERIOD

SELECTION - REPETITION TEST

(1) The five steps in creating programs

- define problem
- design solution (user interface/form settings/flowcharts of events using SmartDraw)
- code solution
 - error handling and debugging
 - types of errors (syntax/run-time/logic)
 - debugging tools and techniques (use UserControlledRepetition)
 - if you get error "Source does not belong to project"
 - delete bin and obj folders and rebuild project
 - stepping (F11) **then** float cursor over var to determine current values
 - setting brkpts
 - documentation (variable dictionaries/program comments)-> demo by adding to Sandwich prg (in Sel-Rep)

=> Practice using SmartDraw

=> Video (csharp10) Methods - use a prep for Next Lesson

(2) Methods

Intro

- Event handler methods vs user defined methods
- Why use -> **songmethod** (demo + then students use this version to create method)
- Using refactoring (right click highlighted code and extract method)

Flowcharting methods

Types of methods

- void type
- type return
- with parameters

Examples

- **simplemethods** (windows app)
- **counter** (console app – note use of static)

=> Methods Assignment (User defined methods) + recall integer arithmetic and modulus operator % (do first question with students)

WORK PERIOD

=> Review (Quiz) then **MathGame**

(3) Simple file concepts

- **SimpleTextFileDemo**
- note: need to add **using System.IO**
- note: when using Application.StartupPath + @ "\ filename" -> file should be in **Debug** folder

Practice Minor Assignment (Work through solution with students - **CreditCardApp**) + video (csharp16 – Textfile)

WORK PERIOD

MAJOR GROUP ASSIGNMENTS

WORK PERIOD

WORK PERIOD

WORK PERIOD

WORK PERIOD

WORK PERIOD

h) **Working with Strings**

(1) Some properties and methods of the string class

- .length
- Convert.ToString() / ToString()
- .StartsWith()
- .EndsWith()
- .IndexOf('c')
- .Substring(-) / Substring(-, -)
- .ToUpper / .ToLower / .Trim() / Replace(-, -)

Examples (optional)

- **StringIntro**
- **Substring**
- **Indexof**
- **Replace**

(2) Sample applications

- Letter and word counting (**LetterCount**) (run from c: drive uses inputbox)

=> Problems 1-3

=> Take up problems

(3) String accumulators using Concatentation

- **StringAccumulators**

=> Problem 2,3

=> Take up problems

(4) Code Validations

- **CreditCard**

=> Product code

Working with Strings Quiz

i) **Arrays**

(1) One Dimensional Arrays

Intro Concepts

- students watch video csharp15 – Arrays – first 9 minutes
- **SimpleArrayIntro**
- demos declaring, storing calculating and displaying arrays
- also features methods with array parameters

Exercise Sheet

Applications

- **ArrayIntermediate**
- Note use of array and variable passing **by ref**
- Further discussion of passing by ref and value -> **passarray**

=> Problem (use ArrayIntermediate and make additions)

=> Take up Problem

Advanced Applications

- intro (pre-amble)
- Array counters (**ArrayCounter**)
- Review (**StudentPoll**)

=> Hand out test review (do sample output question with students they do #2 using ArrayIntermediate as template)

=> Take up Review

One Dimensional Array Quiz

j) **Probabilistic Simulations**

- (1) Intro
- using the Random class
 - three formats
 - i. Next(7) 0-6
 - ii. Next(1,7) 1-6
 - iii. Next(6)+1
 - Demo (**RandomNumberIntro**)
 - Your Turn #1

=> Programming Problems 1,2,3

- (2) Review-Extension
- **MoreRandomNumbers**
 - coin toss problem
 - dice simulation
 - dice counter
 - Sample game (**CardWars**)

=> Gambler Problem

=> Dart Game Problem

=> take up dart game

- (3) Sample applications
- **pinball**

=> Shooting gallery problem (shootgallery)

- (4) Building a discrete random distribution
- intro example (**ballsinnabag**)
 - Extension (**dartboard**)
 - creating a random number generator using a static class

=> Review

PROBABLISTIC SIMULATIONS TEST

k) **Drawing Graphics and Building Games**

(1) **Introduction to the Graphics Class**

- graphics coordinates
- Using the Form Paint Method (*GraphicsClassIntro1*)
 - instantiating a graphics object
 - Font
 - SolidBrush/Brushes
 - DrawString
 - Pen
 - DrawLine /this.width/this.height/ClientRectangle.Width/ClientRectangle.Height
 - DrawRectangle
 - DrawEllipse
 - FillEllipse
 - Loading images from a file (Image.FromFile)/DrawImage
 - StartUpPath not necessary if all images are in Debug folder (this is default path)
 - Dispose

=> Problem

-
- Using a Panel Control (*GraphicsClassIntro2*)
 - panel1.CreateGraphic()/this.CreateGrahics() if using a form
 - using panel.Clear();
 - pens.Color
 - using Color.FromArgb(alpha,r,g,b) + demo (*RGBColors*) note: alpha 0 - invisible 255 solid
 - using width/height / visibility (can layer panels)
 - Demo examples above then work through simple examples with students (paint/panel) then demo *DiceRoll*
 - note: StartupPath can use "\\ or @"\Dice

=> Problem + Additional reading (class8 8.1->8.21)

(2) **Mouse Events**

- Intro Examples
 - *MouseMoveIntro*
 - *MouseHoaveIntro*
 - includes MouseHover/MouseLeave/MouseMove (e.X,e.Y)/MouseDown (e.Button)

=> Minor Assignment (Mouse Problem) Students can run MouseProblem.exe

WORK PERIOD

- Sample Application
 - *blackboard*
 - comment out xLast/yLast lines in MouseMove and observe effects

=> Other things to try (add option to set pen width – use up/down and add option to set background color of blackboard -> *updatedBlackboard*)

(4) **Animation**

- Timers
 - intro (*TimersIntro*)
- Array of Images
 - intro (*ArrayOfImagesDemo*)
 - large arrays (*LogoAnimator*)
 - illustrate extracting frames of an animated GIF using Animation Shop
- Drawing your own animations
 - *FlipBook*

Problem => Extract frames from an animated GIF and animate/draw your own + Additional reading (class10)

WORK PERIOD

- Sample Applications
 - *DiceRollApplication*
 - *SlotMachine*

=> Modify the SlotMachine prg so that it has

- 4 slots
- slows down in two separate stages before reaching the end of the spin
- stop each picture from spinning at different times
- (Hint: use 3 timers – start and finish 1st which then starts 2nd etc , the last timer calls DetermineWins)

Animation Basics

- using `.Top/.Left` (***MovingFlipBook***)
 - note use of 2 timers and wall checking conditions (`this.Width`)
 - experiment with vertical /diagonal movement
 - check for left/right and top/bottom wall hit
 - note problem with `this.height` could use `ClientSize.Height` but then image bounding problem
 - disappearance/reappearance (`PicPlayer.Left = -PicPlayer.Width`)
 - bouncing off the wall (use `dir=10/dir=-dir` and `PicPlayer+=dir`)
- We do ***RocketRace*** together
 - in declaration section** `.left/top`

=> Minor Assignment (Car Race: 2 animating cars race horizontally across the screen)

- Bonus: When user starts race a simulated signal light should display r,y,g. Once g displays race starts

WORK PERIOD

WORK PERIOD

WORK PERIOD

-
- Using `DrawImage()`
 - intro (***DrawImageAnimationIntro***)
 - make sure to set picturebox `sizemode` to `autosize`
 - change background to green and observer problem
 - make image transparent using `GIFCON` (gif only)
 - note key to animation-> `g.Clear(panel1.BackColor)` comment out and observe results
 - Again experiment with
 - vertical, horizontal movement
 - border checking
 - border disappearance (scroll effect)
 - bouncing (***Bouncing***)
 - Image Erasure Technique (***ImageErasure***)
 - Why ? `g.Clear(panel1.BackColor)` vs `FillRectangle()`

=> Problem - Modify `MovingFlipBook` so that it animates using the `ImageErasure` technique

- you will need to convert images to transparent GIFs
- solution (***DrawImageMovingFlipBook***)

Collision Detection and Keyboard Events

- looking at the 4 requirements for collision demo (***CollisionDetectionDemo***)

KeyBoard Events

- control must have **focus** to receive a keyboard event (panel cannot have focus must set focus on form)
- to detect keyboard events you need to set the forms `KeyPreview` to `true`
- The `KeyDown` Event (***KeyBoardDetectionDemo***)
 - the property `e.KeyCode` is used to determine which key is press down
 - `Keys.Right/Keys.Left`

=> Problem (Improve the ***KeyBoardDetectionDemo*** by fully animating bottom character -> ***FullyAnimatedKeyBoardDetection***)

=Extension (***FullAnimKeyBrdDetGrapBkg***) -> no picturebox used /everything is an image object (also using multi-colored bkg)

Sound Effects

- playing sounds (***SoundDemo1***)
 - need -> using `System.Media`
 - add the Windows Media Player (COM component -> Right Click General Items)

Sample Game (***BeachBalls***)/***BeachBallsImageArray***/***BeachBallsImageArrayBkg***

=> Assignment (falling objects game) Individual assignment (partial solution)

WORK PERIOD

WORK PERIOD

WORK PERIOD

WORK PERIOD

WORK PERIOD

WORK PERIOD

FINAL EXAM

Preview of Lego Mindstorms Robotics

NOT A FORMAL CLASS PERIOD

- RCX Internals (intro/overview/hardware) -> graphics.stanford.edu/~kekoa/rcx
 - Rolighed Site -> home14.inet.tele.dk/rolighed (go to Mindstorms Link)
 - Philo Site -> www.philohome.htm/ Legobots at Indiana University -> www.indiana.edu/~legobots
 - Intro to engineering Design (course notes) -> www.owl.net.rice.edu/~elec201/index.html
-

12. ROBOTICS USING LEGOMINDSTORMS

a) Using Lego Mindstorms (Introduction)

- (i) Intro (using Lego CD)
 - Getting started (TOUR)

 - (ii) Setup Part 1 (Part of Training Missions Basic)
INSTALL TOWER FIRST (USB) BEFORE STARTING LEGO PRG (j:\Lego\xppatch)
 - connecting motors
 - connecting sensors
 - touch sensor
 - light sensorSetup Part 2
 - setting up the infra-red transmitter
 - downloading firmware to the RCX

 - (iii) Training Missions (PathFinder) (In **Program RCX** section)
BASIC
 - introduction (building a robot) Roverbot (in Construpedia)
 - creating and downloading a program (RCX code)
 - making changes to your program
 - using touch and light sensors
 - programming your robot to repeat commandsADVANCED
 - small blocks
 - my blocks
 - variables
-

WORK PERIOD

- (iv) Roverbot Project Challenges (Levels 1-3) (A)
 - use tracks for driving base
-

WORK PERIOD

WORK PERIOD

Recap/Preview (Chapt 2 The RCX - Handout) in HelpReference folder

- b) Using Lego Mindstorms (Advanced) Programming with NQC (Not quite C) and the Bricx Command Center (reference:tutorial.doc in HELPPREFERENCE folder)
Note: Use Roverbot with light sensor and bump attachments for lessons 1-14
- (1) The Bricx Command Center (See Help File)
 - starting the command center (choose RCX2)
 - editor window (file-new)
 - loading/editing and saving files (1_simple.nqc) -> in BricxCC\Examples folder
 - using templates (F9)
 - compiling, downloading (**change prg number to 5**) and running
 - direct control tools
 - direct controller
 - brick piano
 - brick joystick
 - diagnostics
 - Redownloading firmware (go to c:\lego\RIS2\script\firmware)
 - (2) Writing your first program in NQC
 - first.nqc
 - task main()/OnFwd()/OnRev()/Wait()/Off
 - OnFwd (turns motor on and sets direction)/Wait is 1/100th's sec
 - debugging
 - changing the speed of the motors
 - firstsetpower.nqc
 - (3) A more interesting program
 - turning1.nqc
 - making turns
 - turning2.nqc
 - defining constants -> #define
 - other examples -> #define leftarm OUT_A ***Very useful
 - repeating programs
 - repeat1.nqc
 - repeat()
 - repeat2.nqc

Students should review and extend the use of today's command (RCS follows a start pattern 4X)

- (4) Using variables
 - spiral.nqc
 - declaring an *int* variable
 - assignment statements/ counter expressions (c += 5;)/i++;
 - random numbers (random.nqc)
 - Random(x) -> 0-x / While(true) -> endless loop
- (5) Control structures
 - if - else statements (if.nqc)
 - note use of == to check for equality/ != not equal/&&/||
 - = is used for assignment statements
 - break (to exit loops) modify random.nqc to illustrate
 - do while (dowhile.nqc)

=> Extend the use of today's commands (TI)

- (6) Sensors (touch/light)
 - Light Sensor
 - line following (linefollower.nqc)
 - using a threshold value
 - Touch Sensor
 - waiting for a sensor (sensorwait.nqc)
 - *until* command
 - avoiding obstacles (sensoravoid.nqc)

Problem:

Use the double bumper attachment that has the two touch sensors and make the robot move away depending on whether the left or right sensor was touched. (doublebumper.nqc)

- (7) Tasks and Subroutines
 - Tasks (multipletasks.nqc)
 - can have at most 10 tasks
 - must have one task with name *main* and this is always executed automatically
 - if you define other tasks you have to explicitly start and stop them
 - running task may start another task - they now run simultaneously
 - Subroutines (subroutine.nqc)
 - use keyword *sub*
 - save memory - stored only once in the RCX
 - you can have 8 subroutines max
 - cannot call subs from other subs
 - can be called from different tasks - but not encouraged
 - Inline Functions
 - waste memory - not stored separately but copied at each place they are used
 - ... *but* no limit to number of inline functions used
 - uses keyword *void* instead of sub
 - Examples
 - inlinefunction1.nqc
 - inlinefunction2.nqc
 - **uses arguments which can be used to pass a value to the function (key advantage)**

Problems:

- modify repeat2.nqc to use subroutine then try to implement a function that uses an argument
- modify if.nqc to use a subroutine
- go through lesson on music from tutorial.doc (see below) *then*
 - modify double bumper problem from yesterday to play two different sounds depending on whether the robot hits the obstacle with the left or right sensor

- (8) Making Music
 - built in sounds
 - playing music
-

- (9) More about motors
 - stopping gently (gentlestoping.nqc)
 - Float() command
 - advanced commands (SetPower/SetDirection/SetOutput)
 - recall SetSensor(SENSOR_2,SENSOR_LIGHT)
- (10) More about sensors
 - sensor type and mode
 - SetSensorType(SENSOR_1,SENSOR_TYPE_LIGHT)
 - SetSensorMode(SENSOR_1,SENSOR_MODE_RAW)
 - there are 8 different modes
 - raw, boolean and percent most common
 - boolean (0,1) default for touch sensor
 - percent default mode for light sensor
 - raw mode outputs numbers in the range 0-1023
 - raw mode specifics
 - touch sensor
 - not touched is 1023
 - fully touched is 50
 - partial touch is 50-1000
 - light sensor
 - 300 is very light
 - 800 is very dark
 - making a proximity sensor (proximity.nqc)
 - makes robot react just before it hits something using light sensor
 - note: attach sensor only not base to input 2
 - place sensor above infra-red port on robot
- (11) Communication between robots
 - SendMessage()/Message()/ClearMessage()
 - Example 1 (slave.nqc/master.nqc)
 - Example 2 (leader.nqc)
- (12) Timers
 - 4 built in timers (Timer(0)...Timer(3))
 - increments in 1/10th's of second/ wait() increments in 1/100's of second
 - Examples (timer1.nqc/timer2.nqc)

Problems:

- add a touch sensor to proximity robot to avoid collisions on the side
- experiment with communication between robots with other groups

=> Project: Bugbot (A)

- need to build Tankbot first (See Handout – Chapt 5 in Help Reference folder)
-

- (13) Debugging Techniques
 - the RCX display (display.nqc)
 - can be used to display values of sensors during prg execution

- SelectDisplay (DISPLAY_SENSOR_1)
- Datalogging (datalog.nqc)
 - the RCX can store values of variables, sensor reading and timers in a piece of memory called the datalog
 - these values can be read by the computer (Tools-Datalog-Upload)
 - CreateDataLog(size of datalog)/AddToDataLog()

- (14) NQC command summary
- NQC Programmers Guide + Help in RCX BrixCC
 - in NQCAPIProgrammersGuide

Review for Test

NQC TEST (K)

=> Go thru RoboticsIntroduction Webpage (www.restena.lu/convict/Jeunes/RoboticsIntro.htm)

- (15) RCX 2.0 Firmware improvements (*OPTIONAL*)
- local variables (local.nqc)
 - SetUserDisplay()
 - SetUserDisplay(source,decimal places)
 - displays source in LCD
 - arrays (average.nqc)
 - PlaySound()
 - parameter can now be a variable - before only constants allowed

WORK PERIOD

WORK PERIOD
