QT Hra – technické detaily projektu

Epic Robot Survival

Autoři: Marek Effenberger, Samuel Hejníček

FIT VUT, ICP

Obecné technické detaily

Projekt je napsán v jazyce C++ ve standardu 17 pomocí Frameworku Qt ve verzi 5.5. Návrh a převážně využívání a chování objektů specifických tomuto prostředí je inspirovaný sérií videí pod tímto odkazem: https://www.youtube.com/watch?v=8ntEQpg7gck&list=PLyb40eoxkelOa5xCB9fvGrkoBf8JzEwtV.

Je dokumentovaný pomocí nástroje Doxygen a obsahuje volně dostupné, případně vygenerované obrázky typu .png jež slouží jako textury. Je přeložitelný pomocí jednoduchého Makefile souboru (dále vizte soubor README).

Koncept a návrh

V této sekci je popsán koncept interakce tříd a myšlenka návrhu hry. Po dané sekci následuje vyobrazení zjednodušeného UML Class diagramu, který hlavně zachycuje dědičnost a vztahy mezi třídami, bez přímé deskripce signál-slotové logiky, jež je více rozepsána v automaticky generované dokumentaci Doxygen.

Koncept

Hlavní kontrolu toku programu má na svědomí třída GameMaster – ta ovládá jednotlivé hlavní tři podcelky: MainWindow, MapWindow a tvorbu samotné scény a hry samotné.

Nejprve je spuštěno MainWindow, to dědí z Qt třídy QMainWindow. Toto okno obsahuje tři možnosti, jak dále ovládat hru. Buď může se zavřít a GameMaster otevře okno MapWindow pro tvorbu nové mapy, nebo může GameMaster dát signál pro otevření již vytvořené mapy ve formátu *JSON*, nebo může ukončit hru.

Okno MapWindow dědí ze třídy QWidget. Tato třída umožňuje uživateli vytvořit mapu, jež je rozprostřena v poli pevně dané velikosti. Pro přehlednost využívá interní třídy pro objekty, které uživatel může rozmístit. Zároveň tvoří instance třídy RobotParamDialog, jež uživateli interaktivně umožňuje zadat parametry robotů.

Výsledný soubor mapy ve formátu *JSON* je buď uložen, nebo rovnou zpracován. Po parsingu tohoto souboru, GameMaster vytvoří scénu a instance tříd jež hru tvoří. Následně je vytvořena instance QGraphicsView jež scénu uživateli vyobrazí.

Jako první je vytvořena jediná instance třídy User. Tato třída pro užívání signál-slotové logiky dědí ze třídy QObject a také ze třídy, pro tvorbu tvaru kolečka, QGrapicsEllipseItem. Tato třída reprezentuje uživatele, a umožňuje ovládání pomocí šipek na klávesnici a interaktivních GUI tlačítek. Při kolizi s jakoukoli instancí třídy Obstacle tvoří User 5 instancí třídy Star, jež kolem svého rodičovského objektu rotuje a na základě interní časové logiky také zanikne. Star je potomek QObject a QGraphicsEllipseItem.

Následně je scéna rozšířena o třídy HorizontalUpperBar a HorizontalLowerBar, obě třídy si tvoří potřebné instance dalších objektů, samy o sobě jsou to jen potomci QGraphicsRectItem a QObject pro tvar obdelníku a jednodušší signál-slotové logiky.

Na nich jsou rozmístěny pomocné objekty třídy Button, jež dědí z třídy QObject a ze třídy QRectItem. Podle instance tato třída umožňuje specifické chování při kliknutí a dále rozšiřují komplexitu signál-slotové interní logiky. Tato třída obsahuje overriding reakce na kliknutí myši a poskytuje vhodný výstup.

Obě třídy HorizontalUpperBar i HorizontalLoweBar mají ukazatel na User instanci, neboť Button instance HorizontalLowerBar uživateli dávají signály k pohybu. Třída HorizontalUpperBar tvoří

instance tříd `Heart`, `GameInfo` a `Timer`. V případě `Heart` se jedná o jednoduchou třídu dědící ze třídy `QObject` a `QGraphicsRectItem`, jež vyobrazuje uživateli počet životů, pomocí signálu od uživatele může měnit svou barvu pro signalizaci ztráty životu. Další jmenované třídy dědí ze tříd `QGraphicsTextItem` a `QObject`. Jedná se o průběžně měněné texty, jež uživatele informují o momentálním stavu hry a časovém limitu pro vítězství. Stejně jako zbytek projektu využívají fontu `Orbitron Extra Bold`, jež se nachází ve složce `Orbitron`.

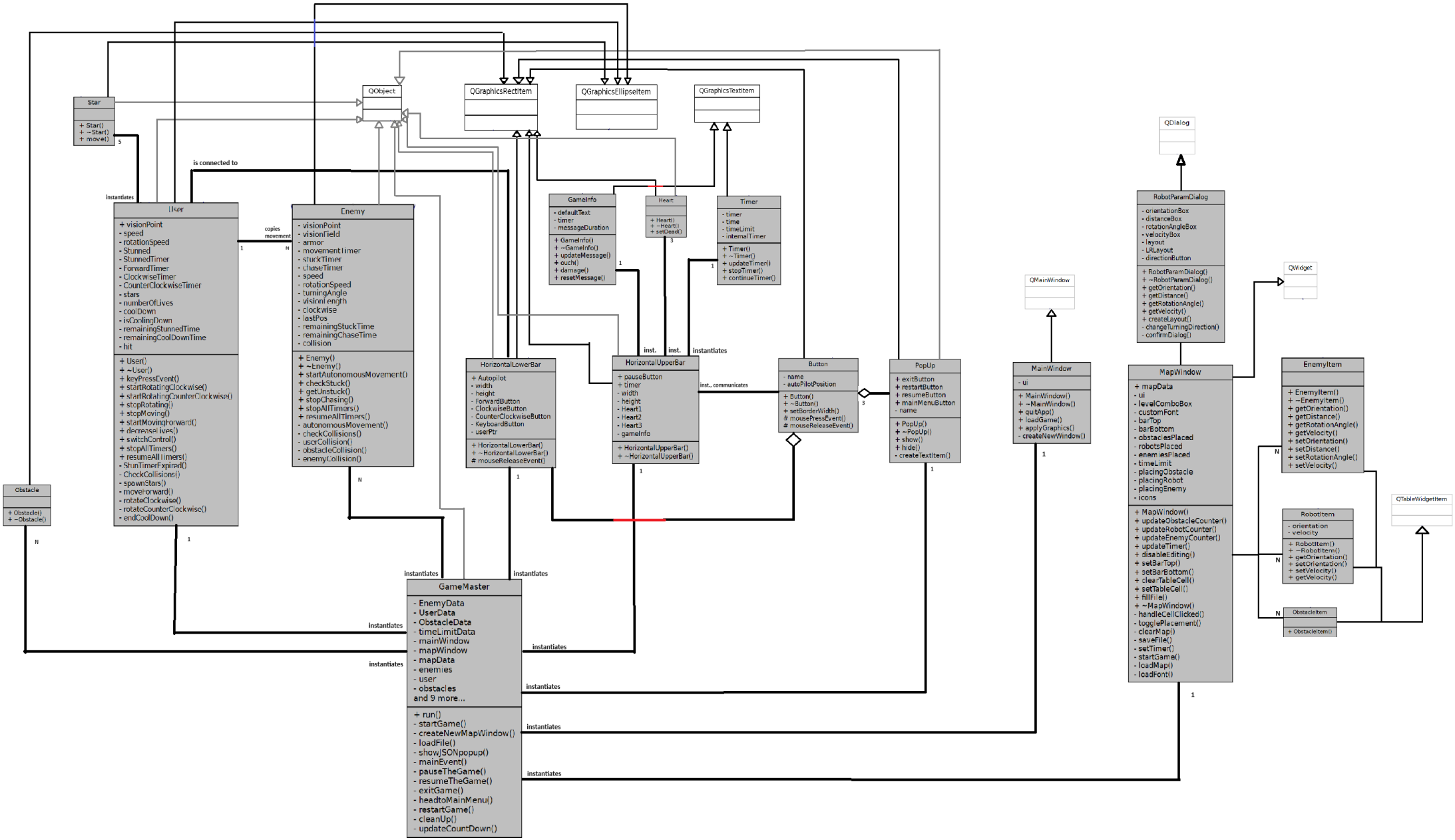
Po tvorbě těchto lišt `GameMaster` vytvoří tři instance třídy `PopUp`. Tato třída dědí z `QGraphicsRectItem` a ze třídy `QObject`. Každý PopUp je tvořen třemi Instancemi třídy `Button`, kdy každý reprezentuje odlišnou logiku a je napojen na příslušné sloty. Efektivně se jedná o interaktivní tabulky bez nutnosti užití `QtCreator`. Instance reprezentují tři možné scénáře hry: Victory, Game Over a Pause. Ihned po vyobrazení scény jsou uživateli skryty, nicméně obsahují sloty, jež je umožní uživateli zaměřit a vyobrazit nad dalšími objekty na scéně.

Instance třídy `Obstacle` vytvoří `GameMaster` na specifikovaných souřadnících, efektivně se jedná pouze o čtverce (potomky třídy `QGraphicsRectItem`) se specifickou strukturou.

Jako poslední jsou vytvořeny instance třídy `Enemy`. Jedná se o autonomní roboty, potomky tříd `QObject` a `QGraphicsEllipseItem`. Každý `Enemy` si drží odkaz na instanci třídy User, pro specifickou logiku trailingu objektu `User`. Výrazným prvkem třídy `Enemy` je daný lichoběžník, jež reprezentuje zorné pole robota a umožňuje mu detekci překážek a následné otáčení. `Enemy` kromě komunikace s `User` objektem, pro logiku ztráty životů a trailingu, taktéž obsahuje rozsáhlou vnitřní logiku, která mu například umožňuje vysmeknutí se zaseknutí mezi překážkami.

Návrh

Na základě signálů GameMaster přepíná mezi třemi zmiňovanými podcelky a efektivně ovládá tvorbu hry. Striktně jsme se při designu hry nedrželi návrhového vzoru, nicméně náš projekt by se dal nejvíce přiblížit vzoru Command, kdy GameMaster slouží jako Command Interface, jež ovládá samotný program, uživatel prostřednictvím vyobrazeného GUI slouží jako Invoker signálů a dle nich GameMaster následně ovládá hru.


Star
+ Star()
+ ~Star()
+ move()
QObject
QGraphicsRectItem
QGraphicsEllipseItem
QGraphicsTextItem
is connected to
instantiates
User
+ visionPoint
- speed
- rotationSpeed
- Stunned
- StunnedTimer
- ForwardTimer
- ClockwiseTimer
- CounterClockwiseTimer
- stars
- numberOfLives
- coolDown
- isCoolingDown
- remainingStunnedTime
- remainingCoolDownTime
- hit
+ User()
+ ~User()
+ keyPressEvent()
+ startRotatingClockwise()
+ startRotatingCounterClockwise()
+ stopRotating()
+ stopMoving()
+ startMovingForward()
+ decreaseLives()
+ switchControl()
+ stopAllTimers()
+ resumeAllTimers()
- ShutTimerExpired()
- CheckCollisions()
- spawnStars()
- moveForward()
- rotateClockwise()
- rotateCounterClockwise()
- endCoolDown()
copies movement
Enemy
- visionPoint
- visionField
- armor
- movementTimer
- stuckTimer
- chaseTimer
- speed
- rotationSpeed
- turningAngle
- visionLength
- clockwise
- lastPos
- remainingStuckTime
- remainingChaseTime
- collision
+ Enemy()
+ ~Enemy()
+ startAutonomousMovement()
+ checkStuck()
+ getUnstuck()
+ stopChasing()
+ stopAllTimers()
+ resumeAllTimers()
- autonomousMovement()
- checkCollisions()
- userCollision()
- obstacleCollision()
- enemyCollision()
GameInfo
- defaultText
- timer
- messageDuration
+ GameInfo()
+ ~GameInfo()
+ updateMessage()
+ ouch()
+ damage()
+ resetMessage()
Heart
+ Heart()
+ ~Heart()
+ setDead()
Timer
- timer
- time
- timeLimit
- internalTimer
+ Timer()
+ ~Timer()
+ updateTimer()
+ stopTimer()
+ continueTimer()
inst.
inst.
instantiates
HorizontalLowerBar
+ Autopilot
- width
- height
- ForwardButton
- ClockwiseButton
- CounterClockwiseButton
- KeyboardButton
- userPtr
+ HorizontalLowerBar()
+ ~HorizontalLowerBar()
# mouseReleaseEvent()
HorizontalUpperBar
+ pauseButton
+ timer
- width
- height
- Heart1
- Heart2
- Heart3
- gameInfo
+ HorizontalUpperBar()
+ ~HorizontalUpperBar()
inst., communicates
Button
- name
- autoPilotPosition
+ Button()
+ ~Button()
+ setBorderWidth()
# mousePressEvent()
# mouseReleaseEvent()
PopUp
+ exitButton
+ restartButton
+ resumeButton
+ mainMenuButton
- name
+ PopUp()
+ ~PopUp()
+ show()
+ hide()
- createTextItem()
Obstacle
+ Obstacle()
+ ~Obstacle()
GameMaster
- EnemyData
- UserData
- ObstacleData
- timeLimitData
- mainWindow
- mapWindow
- mapData
- enemies
- user
- obstacles
and 9 more...
+ run()
- startGame()
- createNewMapWindow()
- loadFile()
- showJSONpopup()
- mainEvent()
- pauseTheGame()
- resumeTheGame()
- exitGame()
- headtoMainMenu()
- restartGame()
- cleanUp()
- updateCountDown()
instantiates
QMainWindow
MainWindow
- ui
+ MainWindow()
+ ~MainWindow()
+ quitApp()
+ loadGame()
+ applyGraphics()
- createNewWindow()
QDialog
RobotParamDialog
- orientationBox
- distanceBox
- rotationAngleBox
- velocityBox
- layout
- LRLayout
- directionButton
+ RobotParamDialog()
+ ~RobotParamDialog()
+ getOrientation()
+ getDistance()
+ getRotationAngle()
+ getVelocity()
+ createLayout()
- changeTurningDirection()
- confirmDialog()
QWidget
MapWindow
+ mapData
- ui
- levelComboBox
- customFont
- barTop
- barBottom
- obstaclesPlaced
- robotsPlaced
- enemiesPlaced
- timeLimit
- placingObstacle
- placingRobot
- placingEnemy
- icons
+ MapWindow()
+ updateObstacleCounter()
+ updateRobotCounter()
+ updateEnemyCounter()
+ updateTimer()
+ disableEditing()
+ setBarTop()
+ setBarBottom()
+ clearTableCell()
+ setTableCell()
+ fillFile()
+ ~MapWindow()
- handleCellClicked()
- togglePlacement()
- clearMap()
- saveFile()
- setTimer()
- startGame()
- loadMap()
- loadFont()
EnemyItem
+ EnemyItem()
+ ~EnemyItem()
+ getOrientation()
+ getDistance()
+ getRotationAngle()
+ getVelocity()
+ setOrientation()
+ setDistance()
+ setRotationAngle()
+ setVelocity()
RobotItem
- orientation
- velocity
+ RobotItem()
+ ~RobotItem()
+ getOrientation()
+ setOrientation()
+ setVelocity()
+ getVelocity()
ObstacleItem
+ ObstacleItem()
QTableWidgetItem