Documentation for Project For Object-oriented Programming #2: Tetris

Bartłomiej Kordek, 14658

User documentation.

1. DVD contents:
    a. Application instalator.
    b. User documentation.
    c. Technical Documentation.
    d. Source code.
2. Installation.
    a. Double click on “setup.exe”.
    b. Click next.
    c. Choose installation folder, click next.
    d. Click Next again and close.
    e. Application should be reachable in selected previously directory.
    f. Some anti virus software may interrupt application process, exclusion should repair it.
3. Application information.
    a. Application can be closed by clicking on x window control.
    b. Application uses hardware acceleration with SDL library, generates randomly elements which are going to be moved to the bottom of window.
4. How to play.
    a. Left cursor – move brick to the left, right cursor – move brick to the right, space – rotate brick.
    b. Close window control – should close window, but for unknown reason application crashes.
5. Uninstallation.
    a. To unninstall application, one have to use windows Programs and Features (select application, click on unninstal).

Technical Documentation

1. Classes informations:

1. Cbrick: This class contains information about part of main grid - brick, such as: brick orientation, brick type.
    1. CoordinatestList GetBlockPositions()const - Stores positions of all slabs from bricks in vector.
    2. void Move(const Direction direction = Direction::D); - Moves brick in selected direction.
    3. virtual void Rotate(const bool clockWise = true) = 0; - Rotates brick.
    4. static void SetBackgroundImage(const Path& path); - Sets background image as string path.
    5. static const String GetImage(); - Return image location as string.
    6. BrickTypes GetBlockType()const; - Returns brick type.
2. CbrickFactory: Mini utility class to generate specific type of brick.
    1. static CBrick* GetBrick(const BrickTypes brickType); - Generates Brick by given enum item.
    2. static CBrick* GetRandomBrick(); - Generates Brick randomly.
3. Cgame: Main game class, manages operation on different objects. This class is singleton.
    1. static CGame* Instance(); - gets pointer to instance.
    2. Static void Destroy(); - destroy instance.
    3. void Initialize(CUInt rowsCount = 50, CUInt columnsCount = 10); - Basic initialization: sets main grid size and sets pictures
    4. void InitWindow(CUInt width = 600, CUInt height = 400); - Initializes main game window.
    5. void ShowGrid(); - Displays main grid.
    6. void MainLoop(); - Main loop method, handles events graphics and logic.
    7. void StartGame(); - Start game method.

8. void SetMainGridBlockBackgroundImage(); - loads block background image.
9. void SetMainGridSlabBackgroundImage(); - oads slab background image.
10. SetWindowColor(); - Sets window color.
11. void AddButton(CUInt x, CUInt y, String name, Path path); - This method adds button with selected position and uses image from Path.
12. void QuitGame(); - quits the game.

4. CmainGrid - This class manages main grid operations.
    1. void SetSize(CUInt rowsCount, CUInt columnsCount, CUInt initialX = 0, CUInt initialY = 0); - Sets size of grid.
    2. void SetBackgroundPicture(const Path& picLocation, CUInt width, CUInt height); - Sets background Picture.
    3. void SetSlabPic(const Path& picLocation, CUInt width, CUInt height) - Sets slab picture.
    4. void AddBrick(const CBrick& brick); - Adds brick to not moving slab pool.
    5. void ReLeaseBrick(); - Creates randomly a brick and puts it on grid.
    6. CUInt GetRowsCount()const; - Returns rows count.
    7. CUInt GetColumnsCount()const; - Returns columns count.
    8. const String SlabPictureLoc()const; - Returns slab picture location.
    9. const String EmptySlabPictureLoc()const; - Returns empty slab picture location.
    10. CUInt GetImgWidth()const; - Returns image width.
    11. CUInt GetImgHeight()const; - Returns image height.

12. CUInt GetSlabRow(CUInt slabIndex)const; - Returns selected slab row.
13. CUInt GetSlabCol(CUInt slabIndex)const; - Returns selected slab column.
14. const bool PartOfSlab(CUInt slabIndex)const -Checks if slab is a part of brick.
15. const bool Empty(CUInt rowIndex, CUInt colIndex)const; - Checks if slab is empty.
16. CUInt SlabCount()const; - Returns slab count.
17. const CoordinatestList ActiveBrickCoords()const; - Gets active brick coordinates.
18. const bool SlabExist(CUInt rowIndex, CUInt colIndex)const - Checks if selected slab exist.
19. void AddCurrentBrickToGrid(); - Converts current brick to main grid slabs.
20. const bool PartOfCurrentBrick(CUInt rowIndex, CUInt colIndex)const; - Checks if selected slab is part of brick.
21. void MoveActualBrick(const Direction direction); - Moves actual brick in selected direction.
22. const bool CheckIfBlockCanBeMoved(const Direction direction)const; - Check if block is movable.
23. void RotateActualBrick(const bool clockWise = true); - Rotates brick.
24. void ManageFullLine(); - Checks for full lines and manages them.

5. Corientation: Class to handle orientation.
    1. void Set(const Direction direction); - Sets orientation.
    2. const Direction Get()const; - Gets orientation.
6. CSDLWrapper: Simple wrapper for SDL graphic library (1.2).

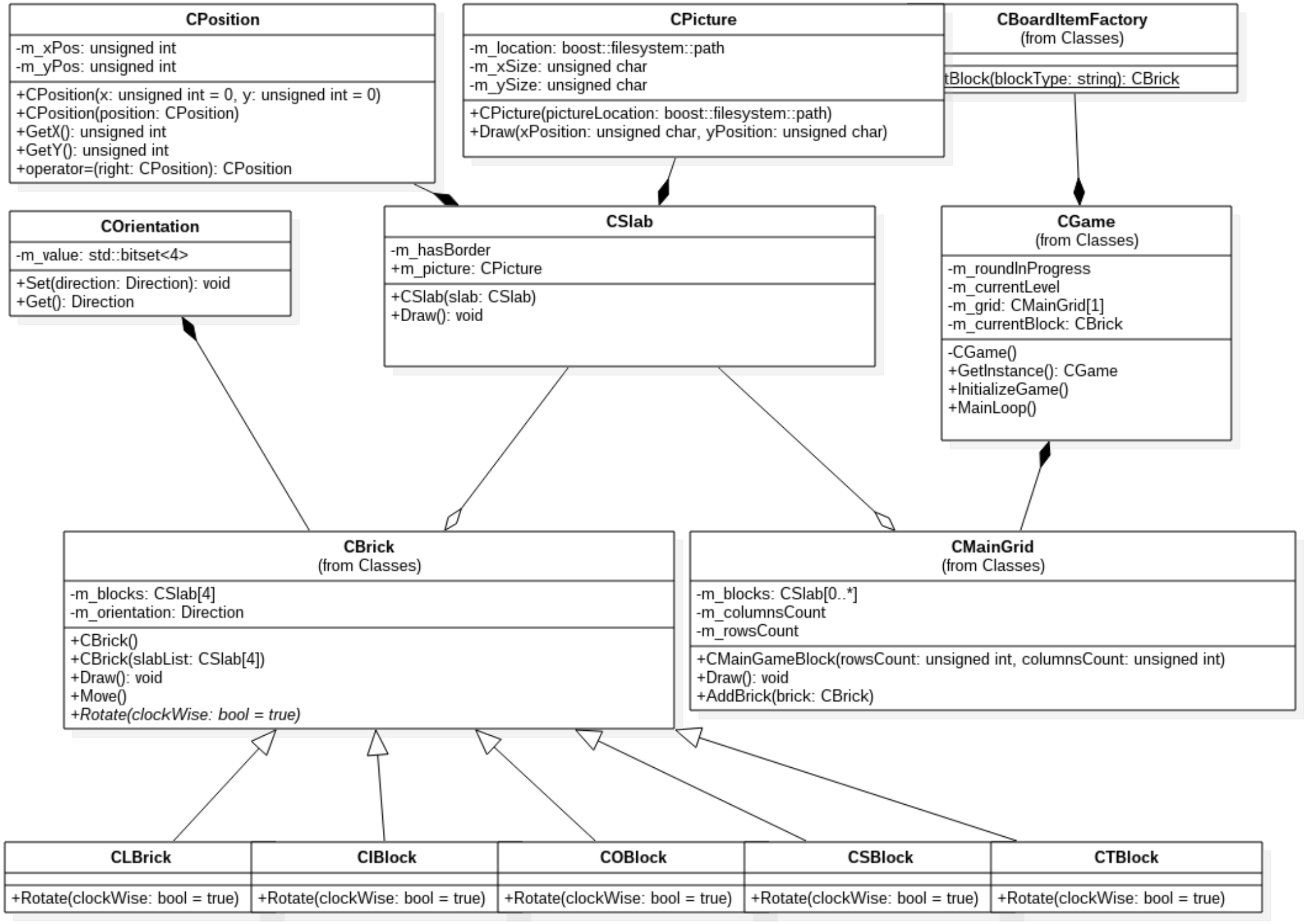
1. void CreateWindow(CUInt width = 0, CUInt height = 0); -
2. void AddImage(const String& path);
3. void ApplyImage(CUInt imgIndex, CUInt x, CUInt y);
4. void Display(const CMainGrid& grid);
5. void Display(const CButton& button);
6. void Actualize();
7. void MainLoop();

7. CtableCoor: This class handles coordinates in table.
    1. CUInt Row()const;
    2. CUInt Col()const;
    3. void Row(CUInt row);
    4. void Col(CUInt col);
    5. void ChangePosition(CUInt row, CUInt col);
    6. TableCoor& operator=(const CTableCoor& coor)

2. Project setup:
    1. To setup project one need to get all sources on choosen directory.
    2. Project has been set to include libraries: boost and SDL, will search for them in disk D:
        1. D:\Libraries\inc – for include files
        2. D:\Libraries\lib – for library files.
        3. D:\Libraries\temp\boost* - for boost library files.
        4. If aboce directories exist with files, project should compile.
        5. Visual Studio 2013 Ultimate 32 was used to develop application and such version is only supported by solution file, older version would not work.
    3. Project is using portable C/C++ libraries and should compile on other O/S (i.e. Linux).
    4. Complete and newest source can be obtained at:

https://github.com/bartekordek/Tetris

3. Classes diagrams.


CPosition
-m_xPos: unsigned int
-m_yPos: unsigned int
+CPosition(x: unsigned int = 0, y: unsigned int = 0)
+CPosition(position: CPosition)
+GetX(): unsigned int
+GetY(): unsigned int
+operator=(right: CPosition): CPosition
CPicture
-m_location: boost::filesystem::path
-m_xSize: unsigned char
-m_ySize: unsigned char
+CPicture(pictureLocation: boost::filesystem::path)
+Draw(xPosition: unsigned char, yPosition: unsigned char)
CBoardItemFactory
(from Classes)
tBlock(blockType: string): CBrick
COrientation
-m_value: std::bitset<4>
+Set(direction: Direction): void
+Get(): Direction
CSlab
-m_hasBorder
+m_picture: CPicture
+CSlab(slab: CSlab)
+Draw(): void
CGame
(from Classes)
-m_roundInProgress
-m_currentLevel
-m_grid: CMainGrid[1]
-m_currentBlock: CBrick
-CGame()
+GetInstance(): CGame
+InitializeGame()
+MainLoop()
CBrick
(from Classes)
-m_blocks: CSlab[4]
-m_orientation: Direction
+CBrick()
+CBrick(slabList: CSlab[4])
+Draw(): void
+Move()
+Rotate(clockWise: bool = true)
CMainGrid
(from Classes)
-m_blocks: CSlab[0..*]
-m_columnsCount
-m_rowsCount
+CMainGameBlock(rowsCount: unsigned int, columnsCount: unsigned int)
+Draw(): void
+AddBrick(brick: CBrick)
CLBrick
+Rotate(clockWise: bool = true)
CIBlock
+Rotate(clockWise: bool = true)
COBlock
+Rotate(clockWise: bool = true)
CSBlock
+Rotate(clockWise: bool = true)
CTBlock
+Rotate(clockWise: bool = true)