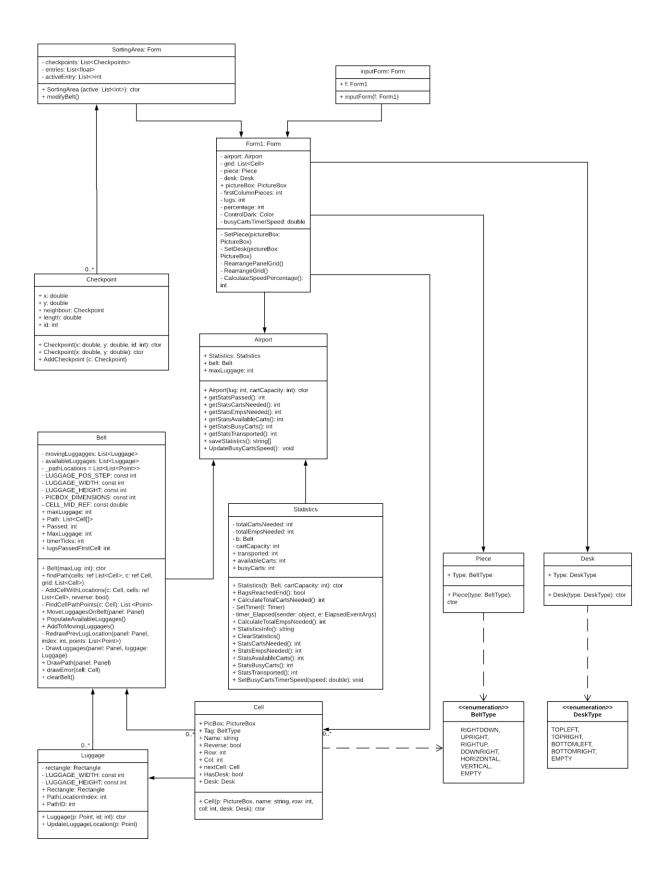# ProCP – Design Document

Artemiy Chervinskyy, 3522644

Robert Andreescu, 3527638

Ana Cirnu, 3644197

Andrei Fruijna, 3524787

Qi-Mo Lin, 3603296

Ignas Apsega, 3620557

SortingArea: Form

- checkpoints: List<Checkpoints>
- entries: List<float>
- activeEntry: List<>int

+ SortingArea (active: List<int>): ctor
+ modifyBelt()

inputForm: Form

+ f: Form1

+ inputForm(f: Form1)

Form1: Form

- airport: Airport
- grid: List<Cell>
- piece: Piece
- desk: Desk
+ pictureBox: PictureBox
- firstColumnPieces: int
- lugs: int
- percentage: int
- ControlDark: Color
- busyCartsTimerSpeed: double

- SetPiece(pictureBox: PictureBox)
- SetDesk(pictureBox: PictureBox)
- RearrangePanelGrid()
- RearrangeGrid()
- CalculateSpeedPercentage(): int

Checkpoint

+ x: double
+ y: double
+ neighbour: Checkpoint
+ length: double
+ id: int

+ Checkpoint(x: double, y: double, id: int): ctor
+ Checkpoint(x: double, y: double): ctor
+ AddCheckpoint (c: Checkpoint)

0..*

Airport

+ Statistics: Statistics
+ belt: Belt
+ maxLuggage: int

+ Airport(lug: int, cartCapacity: int): ctor
+ getStatsPassed(): int
+ getStatsCartsNeeded(): int
+ getStatsEmpsNeeded(): int
+ getStatsAvailableCarts(): int
+ getStatsBusyCarts(): int
+ getStatsTransported(): int
+ saveStatistics(): string[]
+ UpdateBusyCartsSpeed(): void

Belt

- movingLuggagges: List<Luggage>
- availableLuggages: List<Luggage>
- _pathLocations = List<List<Point>>
- LUGGAGE_POS_STEP: const int
- LUGGAGE_WIDTH: const int
- LUGGAGE_HEIGHT: const int
- PICBOX_DIMENSIONS: const int
- CELL_MID_REF: const double
+ maxLuggage: int
+ Path: List<Cell[]>
+ Passed: int
+ MaxLuggage: int
+ timerTicks: int
+ lugsPassedFirstCell: int

+ Belt(maxLug: int): ctor
+ findPath(cells: ref List<Cell>, c: ref Cell, grid: List<Cell>)
- AddCellWithLocations(c: Cell, cells: ref List<Cell>, reverse: bool)
- FindCellPathPoints(c: Cell): List <Point>
- MoveLuggagesOnBelt(panel: Panel)
+ PopulateAvailableLuggages()
+ AddToMovingLuggages()
- RedrawPrevLugLocation(panel: Panel, index: int, points: List<Point>)
- DrawLuggages(panel: Panel, luggage: Luggage)
+ DrawPath(panel: Panel)
+ drawError(cell: Cell)
+ clearBelt()

Statistics

- totalCartsNeeded: int
- totalEmpsNeeded: int
- b: Belt
- cartCapacity: int
- transported: int
+ availableCarts: int
+ busyCarts: int

+ Statistics(b: Belt. cartCapacity: int): ctor
+ BagsReachedEnd(): bool
+ CalculateTotalCartsNeeded(): int
- SetTimer(t: Timer)
- timer_Elapsed(sender: object, e: ElapsedEventArgs)
+ CalculateTotalEmpsNeeded(): int
+ StatisticsInfo(): string
+ ClearStatistics()
+ StatsCartsNeeded(): int
+ StatsEmpsNeeded(): int
+ StatsAvailableCarts(): int
+ StatsBusyCarts(): int
+ StatsTransported(): int
+ SetBusyCartsTimerSpeed(speed: double): void

Piece

+ Type: BeltType

+ Piece(type: BeltType): ctor

Desk

+ Type: DeskType

+ Desk(type: DeskType): ctor

Cell

+ PicBox: PictureBox
+ Tag: BeltType
+ Name: string
+ Reverse: bool
+ Row: int
+ Col: int
+ nextCell: Cell
+ HasDesk: bool
+ Desk: Desk

+ Cell(p: PictureBox, name: string, row: int, col: int, desk: Desk): ctor

0..*

Luggage

- rectangle: Rectangle
- LUGGAGE_WIDTH: const int
- LUGGAGE_HEIGHT: const int
+ Rectangle: Rectangle
+ PathLocationIndex: int
+ PathID: int

+ Luggage(p: Point, id; int): ctor
+ UpdateLuggageLocation(p: Point)

<<enumeration>> BeltType

RIGHTDOWN,
UPRIGHT,
RIGHTUP,
DOWNRIGHT,
HORIZONTAL,
VERTICAL,
EMPTY

<<enumeration>> DeskType

TOPLEFT,
TOPRIGHT,
BOTTOMLEFT,
BOTTOMRIGHT,
EMPTY

Sorting area form

It takes care of finding the shortest path the luggage should take to get to the gate as fast as possible. It is not visible to the user but can be used for improving the quality of the simulation in future updates.

Input form

This form must be filled in for the simulation to start. Here you can input the number of bags that will go on the belts and the capacity of the carts that carry the bags to the planes.

Form1

It allows the user to see the graphical representation of all the actions they can take (the belt, the animation and the statistics are displayed here).

Checkpoint

This is a helper class used to identify the shortest line for a bag within the sorting area out of several randomly created ones.

Airport

This class contains all the information about the belt and the statistics.

Belt

It takes care of all the actions that can be performed on the belt that is built by the user, such as placing check-in desks or animating the process.

Statistics

This class generates the statistics, such as the number of bags taken to the planes, the number of carts used or the amount of employees, needed during the whole simulation.

Cell

This is a helper class containing all the needed information related to one specific instance of a belt. It implements the BeltType enum.

Luggage

It is a helper class containing information about the luggage as a graphical object.

Piece

It is a helper class that allows the user to draw a belt. It implements the BeltType enum.

Desk

This class contains information about the check-in desks. It implements the DeskType enum.