Destroy Around Me

Play in your world game using Image Processing and Computer Vision

# Description

An augmented reality game in which the player interacts with the real world through her mobile device. The game models a physical representation of the scene which the user captures and as the game progress, changes the player’s view of the world using physic to simulate real-world like scene transformation which is augmented on top of the current view the user sees.

The game has an initialization stage in which the user is shown object detected using image processing. When the user is satisfied with the detection, she triggers the beginning of the game at which stage a reference scene is captured, physical model is constructed and in-painting of the original scene is created.

As the game progress, the user interact with the world (both real and virtual) by destroying the objects in it. As the object are destroyed, the physical model is changed and the virtual scene is changed in accordance and augmented into the mobile device view the user sees.

# Game Design

Following section briefly describes the game design and the code use for implementation

## Overview

The game consists of… using open CV and xyz frameworks

## Code

Following you will find description of the code and sequences used in the game

### Classes

Following is a high level class diagram of the important components involved in the game

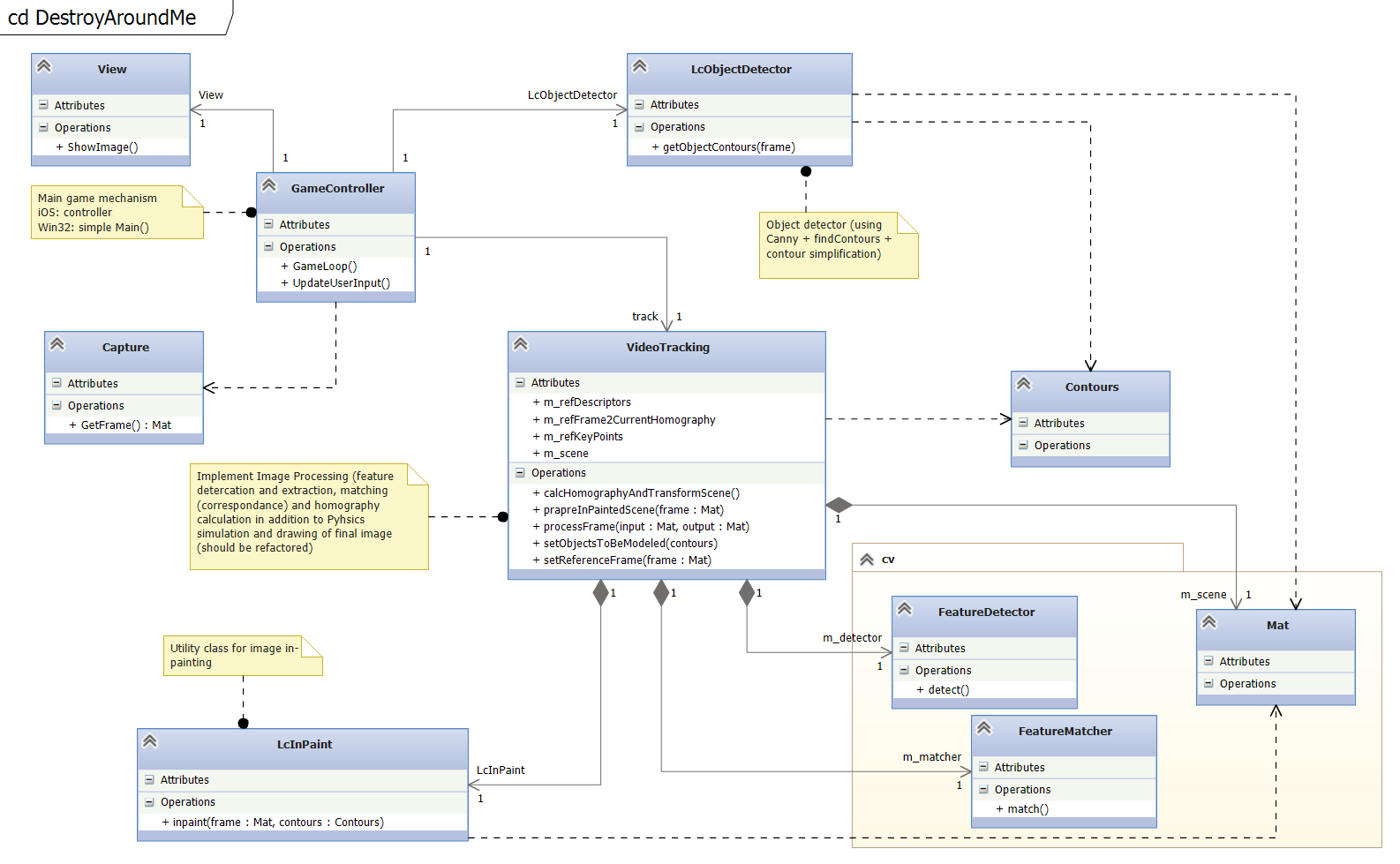


Figure 1: Class diagram

Following are high level descriptions of the various classes depicted in ***Figure 1: Class diagram***.

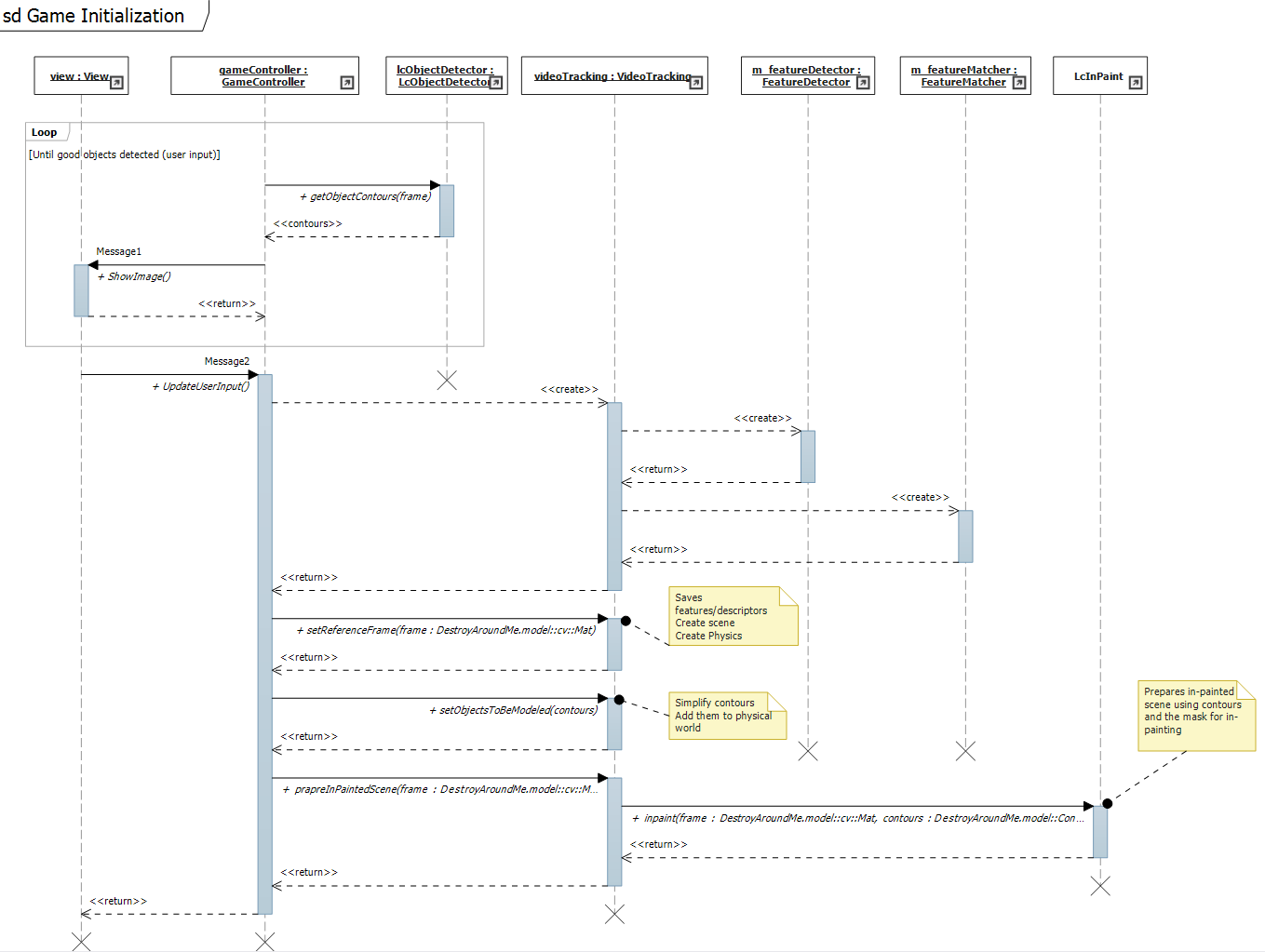
|  |  |
| --- | --- |
| Class | Description |
| GameController | The game controller running the game loop and using other classes to facilitate the actual game |
| View/Capture | External classes (platform dependent) that the GameController uses for capturing images/video and display the results. |
| LcObjectDetector | A canny bases object detector that is used to identify objects (edges) in the scene and calculate contours |
| LcInpaint | A simple wrapper class used by VideoTracking for in-painting using OpenCV inpaint() method. The class takes contours found by LcObjectDetector, offsets them and then uses them to create a mask for in-paint function. The results are saved in a member field of VideoTracking and uses to in-paint “destroyed” parts in the scene. |
| VideoTracking | The main class of the game. Used by controller to process new frame, finding features, matching them with reference features, calculates homography and transform the scene, augmenting to it the current scene state (e.g. ball, barriers (user touches) and destroyed parts). |
| Contours | Abstract notion of contours – a list of edges (either in Open CV format or other library formats (e.g. physics body fixtures). |
| Mat | Open CV matrix – used for image manipulation |
| FeatureDetector | An abstract notion of OpenCV feature detector/descriptor extractor. Currently implemented as ORB feature detector |
| FeatureMatcher | An abstract notion of OpenCV feature/descriptor matcher. Currently using FlannBasedMatcher matcher with location sensitive hash algorithm. |

### Sequences

Following are sequence diagrams describing the high level flow of the game execution

#### Game Initialization

The game initialization sequence happens at the beginning of the game. In this sequence the user (player) is displayed with a preview of the object detection results and when satisfied with the results triggers the beginning of the game which involves in the actual construction of the augmentation scene and physical model of the world in his view



#### Game Loop

The game loop is the actual game sequence in which the physical worlds in updated and the augmentation of the calculated transformed scene is shown to the user.

