**ImageScaling using Bilinear Interpolation and Trilinear Filtering**

Algorithm:

1. Open source png file and load data into memory
2. Get scaling input (scaleX, scaleY) from user
3. Generate two mipmaps (large = desired image x 1.1 and small = desired image x 0.9) from Source image data using bilinear interpolation
4. Use trilinear filtering to obtain desired image
   1. Downsize bigger mipmap using bilinear interpolation
   2. Upsize smaller mipmap using bilinear interpolation
   3. Linear interpolate between two images to get output image data
5. Write processed image data to png output file.

Sample Example:

|  |
| --- |
| 100  Actual Output  Bilinear Interpolation  Trilinear Filtering  DownSize  Bigger Mipmap  45  0.9x  1.1x  55  55  LinearInterpolate  50x50  50x50  50  50  100  Smaller Mipmap (Dest x 0.9)  Bigger Mipmap (Dest x 1.1)  UpSize  Smaller Mipmap  50x50  Output Image Data  Source Image Data  45  64  64  Output Image Data  Source Image Data |

**Class Diagrams:**

|  |
| --- |
| **CBilinearTask**  bilinearInterpolation:mfunc  //+printResult()  **CBilinearTaskDataIn**  const unsigned char\*\*: source\_data  int: w\_source, h\_source,  start\_height, end\_height,  w\_dest  unsigned char\*\* dest\_data  Point2D: destPoint;  Float:scaleX scaleY;  **CLinearTaskDataIn**  const unsigned char\*\*:  bi\_high\_rezied, bi\_low\_rezied;  unsigned char\*\*:  dest\_data  int: start\_height, end\_height, w\_dest;  float:h3;  **CLinearTask**  linearInterpolat: mfunc  //+printResult()  **ITask**  void\*:mTaskDataIn;  void\*:mTaskResultOut;  unsigned int:mTaskId;  +submitTask(ITask \*task);  +startProcess();  +clearAllTask();  +maxTaskCount();  **TaskProcessor**  **IState**\*:mStatus  vector<ITask\*>:mTaskList  vector<HANDLE>:mThreadHandle  +submitTask(ITask \*task);  +startProcess();  +clearAllTask();  +maxTaskCount(); |

|  |
| --- |
| **STATUS\_READY**  **STATUS\_COMPLETED**  **STATUS\_IS\_IN\_PROGRESS**  **STATUS\_NO\_TASK**  **IState**  +submitTask()  +startProcess()  +clearAllTask() |

**Multithreading:**

Image\_library uses threads to perform pixels operations. Details below:

1. Library receives a chunk of data (image pixels)
2. Initiates a TaskProcessor and gets maximum number of threads available.
3. Divides memory pixels into chunk of data and issue(submits) each chunk as a task to execute on TaskProcessor
4. TaskProcessor executes all tasks in threads and waits till all threads are completed.
5. Libraries get the out data pointer as processed pixels

Bilinear Interpolation and Trilinear Filtering both library methods uses the above technique to process pixels, which maximizes the performance.