



Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

## **ОТЧЕТ ПО (УЧЕБНОЙ/ПРОИЗВОДСТВЕННОЙ)** **ПРАКТИКЕ**

Студент Брянская Екатерина Вадимовна

*фамилия, имя, отчество*

Группа ИУ7-42Б

Тип практики учебная

Название предприятия \_\_\_\_\_

Студент

Брянская Е.В.

*подпись, дата*

*фамилия, и.о.*

Руководитель практики

*подпись, дата*

*фамилия, и.о.*

Оценка \_\_\_\_\_

2020 г.

## Оглавление

Введение.....	4
1. Аналитический раздел .....	5
1.1 Объекты сцены.....	5
1.2 Анализ алгоритмов удаления невидимых линий и поверхностей .....	6
1.2.1 Алгоритм Робертса.....	7
1.2.2 Алгоритм, использующий z-буфер.....	7
1.2.3 Алгоритм обратной трассировки лучей .....	8
1.2.4 Алгоритм Варнока.....	8
1.2.5 Вывод.....	9
1.3 Анализ алгоритмов закраски .....	9
1.3.1 Простая закрашка .....	9
1.3.2 Закраска методом Гуро .....	9
1.3.3 Закраска Фонга.....	10
1.3.4 Вывод.....	10
1.4 Анализ алгоритмов построения теней.....	10
1.5 Анализ моделей освещения .....	11
1.5.1 Модель Ламберта .....	11
1.5.2 Модель Фонга.....	11
1.5.3 Модель Блинна-Фонга.....	12
1.5.4 Вывод.....	12
1.6 Анализ алгоритмов наложения текстур на объекты трехмерной сцены .....	13
1.7 Физическая модель поведения объектов текущей сцены.....	13
1.8 Вывод из аналитической части.....	13
2. Конструкторский раздел.....	15
2.1 Возможности программы.....	15
2.2 Общий алгоритм работы программы .....	15
2.3 Создание полигональной сетки .....	17
2.4 Алгоритм z-буфера.....	18
2.5 Алгоритм закраски по Гуро .....	19
2.6 Модель освещения Фонга.....	19
2.7 Физика песка .....	21
2.8 Используемые типы и структуры данных .....	23
3. Технологический раздел.....	26
3.1 Выбор и обоснование языка программирования и среды разработки .....	26

___3.2	Формат входных данных.....	26
___3.3	Структура и состав классов .....	27
___3.4	Предоставляемый интерфейс.....	27
4.	Исследовательский раздел.....	28
	Заключение.....	29
	Список использованной литературы.....	30
	Приложение.....	31

## **Введение**

В современном мире роль компьютерной графики огромна, она применима везде, где нужно создание и обработка изображений и каких-либо цифровых данных. Спецэффекты в фильмах, реклама, компьютерные игры – всё это проявления компьютерной графики, которая используется повсеместно. Она применяется в большинстве инженерных и научных дисциплинах для передачи информации, её наглядного восприятия.

Одной из основных задач компьютерной графики является создание наиболее приближенного к реальной жизни изображения. В процессе работы необходимо учитывать свойства не только самого объекта, но и окружающих тел, среды. Существует много алгоритмов, которые решают данную задачу, но зачастую они достаточно затратны как по времени, так и по используемой памяти. Поэтому необходимо корректно подбирать алгоритм при моделировании чего-либо, исходя из конкретной цели.

Цель данной работы - разработать программу моделирования работы песочных часов.

Измерительное время пользователь выбирает из предлагаемых вариантов. Программа даёт возможность задания в соответствующем текстовом поле положения источника освещения, камеры, её перемещения вокруг объекта. Пользователь может начать повторный отсчет по истечении времени работы прибора. Программа должна предусматривать построение теней часов и песка в них. При разработке программы необходимо учитывать прозрачность стекла и блики от источников освещения.

В ходе достижения поставленной цели решается следующий ряд задач:

- 1) описать из каких объектов состоит сцена, как они представляются и как будут заданы в программе
- 2) проанализировать алгоритмы для визуализации трёхмерной сцены, при необходимости рассмотреть модификации, и сделать выбор

- 3) разработать физическую модель песочных часов, которая была бы наиболее приближенной к реальности
- 4) реализовать выбранные алгоритмы
- 5) реализовать интерфейс, отвечающий заданным требованиям и предоставляющий пользователю заявленные возможности

## **1. Аналитический раздел**

### **1.1 Объекты сцены**

Сцена состоит из следующих объектов:

- непосредственно ограничивающая плоскость – расположена параллельно плоскости  $OXZ$
- песочные часы – располагаются на ограничивающей плоскости

Можно выделить следующие составляющие:

1. стеклянная часть – её свойства необходимо учитывать при изображении теней, бликов, моделировании песка
  2. песок – совокупность мелких песчинок, которая приходит в движение при отсчете времени. Время вышло – состояние, при котором всё содержимое верхней части часов пересыпалось в нижнюю.
  3. непрозрачные подставки (сверху и снизу) – представляют собой параллелепипеды, основания которых параллельны ограничивающей плоскости, а боковые ребра перпендикулярны ей
- источник освещения – начальное положение указывается по умолчанию, но пользователь может его поменять. Но предполагается, что источник находится в бесконечности
  - камера - начальное положение указывается по умолчанию, но пользователь может его изменить, указав координаты в соответствующие текстовые поля

Непрозрачные подставки, стеклянная часть часов будут описываться с помощью каркасной модели, так как любые другие будут содержать информацию, которая будет излишней для данной задачи.

Песок “разбивается” на части:

- песок, который ещё не пересыпался, и песок, который уже пересыпался – будут представляться в виде полигональной сетки
- движущиеся частички – представляются в виде каркаса треугольных пирамид

Использование объёмной модели в данной задаче не рационально в силу излишней информации для данных объектов и затрат по памяти.

## **1.2 Анализ алгоритмов удаления невидимых линий и поверхностей**

Задача удаления невидимых линий и поверхностей является достаточно сложной, поэтому существует много алгоритмов, решающих её. Но наилучшего решения общей задачи удаления нет, поэтому нужно ориентироваться на цель, преследуемую в конкретной работе.

Одной из главных задач состоит в том, чтобы наиболее правдоподобно показать пересыпание песка, так как основная идея песочных часов заключается именно в этом. Для этого нужно добиться плавности движения и равномерности движения частиц, то есть важно качество анимации. Другими словами, главным фактором является быстроедействие.

Рассмотрим некоторые из основных алгоритмов удаления невидимых линий и поверхностей, проанализируем их и выберем наиболее подходящий.

### **1.2.1 Алгоритм Робертса**

Плюсом данного алгоритма является то, что математические методы, которые здесь задействованы, являются достаточно мощными и точными, а сам алгоритм прост для понимания.

На первом этапе удаляются те рёбра или грани, которые экранируются самим телом, далее те, что экранируются другими телами сцены. И если тела, связаны отношением взаимного протыкания, то удаляются невидимые линии пересечения тел.

К минусам алгоритма следует отнести тот факт, что вычислительная ёмкость растёт как квадрат числа объектов. То есть при большом количестве объектов на сцене алгоритм не будет отличаться своим быстрым действием. Но существуют различные оптимизации, например, с использованием предварительной приоритетной сортировки вдоль оси  $z$ [1].

### **1.2.2 Алгоритм, использующий z-буфер**

К достоинствам можно отнести то, что это один из простейших алгоритмов удаления невидимых линий и поверхностей. Также он позволяет визуализировать пересечения этих поверхностей. Элементы сцены не нужно сортировать, следовательно, на это не тратится время.

Сцена может быть любой сложности, а так как размеры изображения ограничены размером экрана, то трудоёмкость линейно зависит от числа поверхностей.

Но есть и существенные недостатки у этого алгоритма: большой объём требуемой памяти, трудоёмкость реализации таких эффектов, как прозрачность и освещение, но есть специальные модификации, направленные на разрешение проблемы с их визуализацией.

### **1.2.3 Алгоритм обратной трассировки лучей**

В этом алгоритме считается, что точка зрения находится в бесконечности на положительной полуоси  $z$  и поэтому все световые лучи параллельны этой оси. Траектория каждого луча отслеживается, чтобы определить, какие именно объекты сцены, если таковые существуют, пересекаются с данным лучом.

Этот алгоритм позволяет получать такие эффекты как отражение, преломление и т.д. Другими словами, изображение становится более реалистичным. Расчёт отдельной точки изображения производится независимо от других, поэтому в этом алгоритме поддерживается высокая степень параллельности вычислений.

К недостаткам этого алгоритма нужно отнести высокую вычислительную стоимость расчетов. Каждая точка изображения требует много вычислительных операций. Очень много времени тратится на поиск пересечений, что зачастую тормозит работы алгоритма[1].

### **1.2.4 Алгоритм Варнока**

Единой версии этого алгоритма не существует. Но все модификации основываются на рекурсивном разбиении окна.

Для каждого окна определяются многоугольники, которые связаны с ним, и те, у которых легко определить видимость, изображаются. Если нет, то разбиение на подокна продолжается до тех пор, пока либо нельзя будет принять однозначное решение, либо размер окна не станет равным одному пикселю.

На каждом этапе разбиения осуществляется определение расположения многоугольников относительно текущего окна.

Насколько быстро будет работать алгоритм, зависит от сложности сцены. Так как было решено использовать полигональную сетку, то это может затормозить выполнение алгоритма.



### **1.2.5 Вывод**

Проанализировав алгоритмы удаления невидимых линий и поверхностей, можно подобрать наиболее подходящий для данной задачи.

В силу того, что в сцене будет присутствовать такой сложный объект, как песок, который будет пересыпаться, использование алгоритма Варнока нецелесообразно, так как сцена будет достаточно сложной.

Как уже отмечалось алгоритм обратной трассировки хоть и даёт реалистичное изображение, но требует больших временных затрат, в текущей же задаче быстроедействие алгоритма играет одну из ключевых ролей.

Предпочтительнее использовать алгоритм с z-буфером, так как он позволяет работать со сценой любой сложности и обеспечивает её обработку за меньшее время. Кроме того, при работе с освещением, закраской z-буфер легко подстраивается под модификации.

## **1.3 Анализ алгоритмов закраски**

### **1.3.1 Простая закрашка**

Один из самых быстрых алгоритмов. В основе лежит закон Ламберта, который говорит о том, что интенсивность отражённого света пропорциональна косинусу угла между направлением света и нормалью к поверхности.

Поверхность предметов, которая была изображена с помощью этого алгоритма, выглядит визуально матовой и блеклой. Это происходит потому, что вся грань закрашивается с постоянной интенсивностью [1].

### **1.3.2 Закраска методом Гуро**

Благодаря этому алгоритму можно получить сглаженное изображение. Сначала определяется интенсивность вершин, а затем с помощью билинейной интерполяции вычисляется интенсивность соответствующего пикселя.

Недостаток алгоритма – появления полос Маха, это можно исправить, если увеличить число обрабатываемых граней, но это приводит к замедлению процесса визуализации [3].

Закраску Гуро лучше всего использовать с простой моделью с диффузным отражением, так получается более реалистичное изображение [1].

### **1.3.3 Закраска Фонга**

В закраске по Фонгу интерполирование происходит по вектору нормали, в отличие от закраски по Гуро, где используется значение интенсивности. Тем самым изображение становится более реалистичным, а зеркальные блики выглядят более правдоподобно.

Но у этого метода есть существенный недостаток: большие вычислительные затраты, так как сначала производится работа с нормальями, а затем по ним находятся соответствующие интенсивности [1].

### **1.3.4 Вывод**

Так как в текущей задаче будут использоваться полигональные сетки и, желательно, чтобы были незаметны, сглажены границы многоугольников, то лучше всего подойдёт алгоритм закраски по Гуро.

## **1.4 Анализ алгоритмов построения теней**

Для придания естественности необходимо учитывать тени от объектов. Из всех алгоритмов, выполняющих данную задачу, нужно выбрать наиболее подходящий для реализации текущей трёхмерной сцены.

В общем случае построение теней производится в два этапа удаления невидимых поверхностей (относительно положения наблюдателя и каждого

источника освещения). Поэтому построение теней лучше всего внедрить в выбранный алгоритм удаления невидимых поверхностей [1].

При построении теней может использоваться матрица теней, в которой помечаются затеняемые многоугольники и многоугольники, отбрасывающие тень.

Определять видимость объектов сцены можно с помощью алгоритмов z-буфера, Робертса и других. Удобнее работать с алгоритмом z-буфера, так как он позволит сделать это быстрее.

Важно помнить тот факт, что положение теней зависит только от расположения источника, а не от наблюдателя.

## **1.5 Анализ моделей освещения**

Освещение тоже играет не последнюю роль в данной задаче, а в силу того, что на сцене будет моделироваться стекло, нужно подобрать соответствующую модель.

### **1.5.1 Модель Ламберта**

Является одной из самых простых моделей освещения. Она моделирует идеальное диффузное освещение, сама поверхность выглядит одинаково яркой со всех направлений. Считается, что при попадании на поверхность свет равномерно рассеивается по всем направлениям. В основе расчетов лежит закон Ламберта.

### **1.5.2 Модель Фонга**

Представляет из себя классическую модель освещения, комбинируя диффузную составляющую (модель Ламберта) и зеркальную составляющую. Таким образом, на объекте может появиться ещё и блик, что придаёт больше

Модель Фонга учитывает только свойства текущей точки и источника освещения, а такие эффекты как рассеивание, отражение от других тел игнорируются.

Есть большое сходство с моделью Фонга, только она исключает расчёт отражённого луча, что гораздо упрощает вычисления, тем самым, экономится время работы. Но существенной разницы нет.

Чем ближе такой вектор к нормали поверхности, тем больше будет вклад зеркальной компоненты.

### 1.5.4 Вывод

```
#####
#####
#####
```

1  
 2  
 3  
 4  
 5  
 6  
 7  
 8  
 9  
 10  
 11  
 12  
 13  
 14  
 15  
 16  
 17  
 18  
 19  
 20  
 21  
 22  
 23  
 24  
 25  
 26  
 27  
 28  
 29  
 30  
 31  
 32  
 33  
 34  
 35  
 36  
 37  
 38  
 39  
 40  
 41  
 42  
 43  
 44  
 45  
 46  
 47  
 48  
 49  
 50  
 51  
 52  
 53  
 54  
 55  
 56  
 57  
 58  
 59  
 60  
 61  
 62  
 63  
 64  
 65  
 66  
 67  
 68  
 69  
 70  
 71  
 72  
 73  
 74  
 75  
 76  
 77  
 78  
 79  
 80  
 81  
 82  
 83  
 84  
 85  
 86  
 87  
 88  
 89  
 90  
 91  
 92  
 93  
 94  
 95  
 96  
 97  
 98  
 99  
 100  
 101  
 102  
 103  
 104  
 105  
 106  
 107  
 108  
 109  
 110  
 111  
 112  
 113  
 114  
 115  
 116  
 117  
 118  
 119  
 120  
 121  
 122  
 123  
 124  
 125  
 126  
 127  
 128  
 129  
 130  
 131  
 132  
 133  
 134  
 135  
 136  
 137  
 138  
 139  
 140  
 141  
 142  
 143  
 144  
 145  
 146  
 147  
 148  
 149  
 150  
 151  
 152  
 153  
 154  
 155  
 156  
 157  
 158  
 159  
 160  
 161  
 162  
 163  
 164  
 165  
 166  
 167  
 168  
 169  
 170  
 171  
 172  
 173  
 174  
 175  
 176  
 177  
 178  
 179  
 180  
 181  
 182  
 183  
 184  
 185  
 186  
 187  
 188  
 189  
 190  
 191  
 192  
 193  
 194  
 195  
 196  
 197  
 198  
 199  
 200  
 201  
 202  
 203  
 204  
 205  
 206  
 207  
 208  
 209  
 210  
 211  
 212  
 213  
 214  
 215  
 216  
 217  
 218  
 219  
 220  
 221  
 222  
 223  
 224  
 225  
 226  
 227  
 228  
 229  
 230  
 231  
 232  
 233  
 234  
 235  
 236  
 237  
 238  
 239  
 240  
 241  
 242  
 243  
 244  
 245  
 246  
 247  
 248  
 249  
 250  
 251  
 252  
 253  
 254  
 255  
 256  
 257  
 258  
 259  
 260  
 261  
 262  
 263  
 264  
 265  
 266  
 267  
 268  
 269  
 270  
 271  
 272  
 273  
 274  
 275  
 276  
 277  
 278  
 279  
 280  
 281  
 282  
 283  
 284  
 285  
 286  
 287  
 288  
 289  
 290  
 291  
 292  
 293  
 294  
 295  
 296  
 297  
 298  
 299  
 300  
 301  
 302  
 303  
 304  
 305  
 306  
 307  
 308  
 309  
 310  
 311  
 312  
 313  
 314  
 315  
 316  
 317  
 318  
 319  
 320  
 321  
 322  
 323  
 324  
 325  
 326  
 327  
 328  
 329  
 330  
 331  
 332  
 333  
 334  
 335  
 336  
 337  
 338  
 339  
 340  
 341  
 342  
 343  
 344  
 345  
 346  
 347  
 348  
 349  
 350  
 351  
 352  
 353  
 354  
 355  
 356  
 357  
 358  
 359  
 360  
 361  
 362  
 363  
 364  
 365  
 366  
 367  
 368  
 369  
 370  
 371  
 372  
 373  
 374  
 375  
 376  
 377  
 378  
 379  
 380  
 381  
 382  
 383  
 384  
 385  
 386  
 387  
 388  
 389  
 390  
 391  
 392  
 393  
 394  
 395  
 396  
 397  
 398  
 399  
 400  
 401  
 402  
 403  
 404  
 405  
 406  
 407  
 408  
 409  
 410  
 411  
 412  
 413  
 414  
 415  
 416  
 417  
 418  
 419  
 420  
 421  
 422  
 423  
 424  
 425  
 426  
 427  
 428  
 429  
 430  
 431  
 432  
 433  
 434  
 435  
 436  
 437  
 438  
 439  
 440  
 441  
 442  
 443  
 444  
 445  
 446  
 447  
 448  
 449  
 450  
 451  
 452  
 453  
 454  
 455  
 456  
 457  
 458  
 459  
 460  
 461  
 462  
 463  
 464  
 465  
 466  
 467  
 468  
 469  
 470  
 471  
 472  
 473  
 474  
 475  
 476  
 477  
 478  
 479  
 480  
 481  
 482  
 483  
 484  
 485  
 486  
 487  
 488  
 489  
 490  
 491  
 492  
 493  
 494  
 495  
 496  
 497  
 498  
 499  
 500  
 501  
 502  
 503  
 504  
 505  
 506  
 507  
 508  
 509  
 510  
 511  
 512  
 513  
 514  
 515  
 516  
 517  
 518  
 519  
 520  
 521  
 522  
 523  
 524  
 525

Как было уже упомянуто, основной целью является моделирование реалистичного процесса работы песочных часов. Вместе с этим большое внимание уделяется эффективности задействованных алгоритмов. Задача непростая, так как песок будет состоять из большого числа частиц, которые должны равномерно пересыпаться, каждая из них обладает скоростью, размером и т.д.

Начальное положение часов: весь песок находится в верхней части. Далее с течением времени он пересыпается в нижнюю. То есть, объём песка в верхней части должен равномерно уменьшаться, причём должны соблюдаться соответствующие физические законы. Аналогичная картина будет наблюдаться в нижней части, только количество песка должно увеличиваться. При моделировании теней необходимо учитывать тот факт, что и движущийся песок отбрасывает тень, доля анимации должна присутствовать и здесь. Время пересыпания должно соответствовать реальному.

Вся система осложняется наличием стеклянной части, свойства которой необходимо учитывать при освещении, моделировании теней и т.д.

В данном разделе были рассмотрены алгоритмы удаления невидимых линий и поверхностей, закраски, алгоритмы построения теней, проанализирована физическая модель песочных часов.

В результате анализа для моделирования был выбран алгоритм z-буфера (удаление невидимых линий и поверхностей), метод Гуро (закраска), построение теней будет осуществляться с помощью z-буфера, а песок будет представляться в виде системы частиц, которая подчиняется соответствующим физическим законам.

Входным данным является измерительное время, которое пользователь выбирает из предлагаемого списка. Пользователь может поменять расположение камеры, источника освещения. Все изменения, которые могут быть внесены пользователем, должны быть учтены при разработке графического интерфейса программного продукта.

## **2. Конструкторский раздел**

В этом разделе будут описаны используемые методы и алгоритмы.

### **2.1 Возможности программы**

Программа предоставляет возможность:

- выбора из предлагаемого списка измерительного времени
- задания в соответствующем текстовом поле положения источника освещения, камеры
- перемещать камеру вокруг объекта
- повторного отсчета по истечении времени работы прибора

### **2.2 Общий алгоритм работы программы**

Общий алгоритм выглядит следующим образом (Рисунок 1):

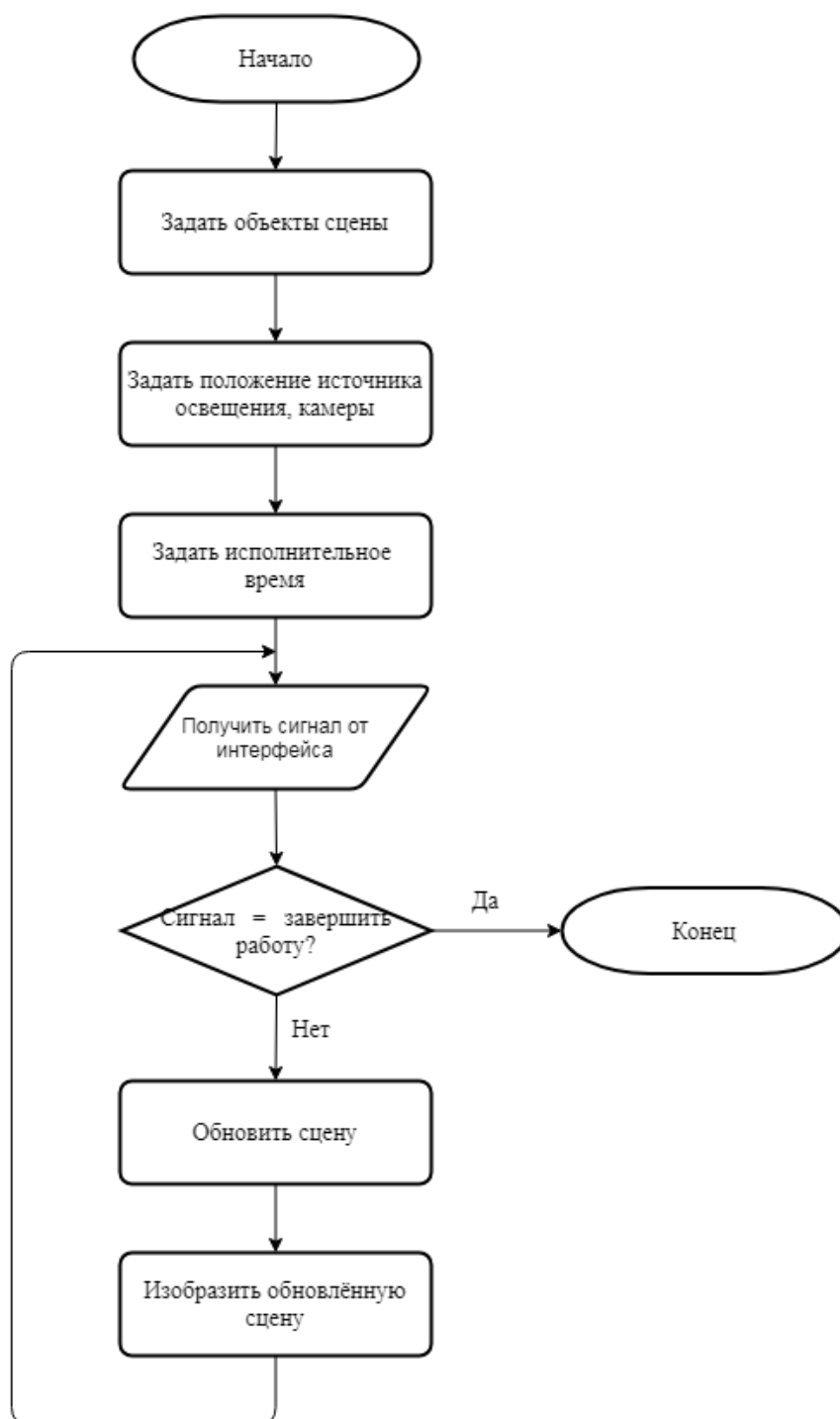


Рисунок 1 - Общий алгоритм работы

В качестве сигнала может выступать команда перемещения камеры вокруг объекта, смены положения источника освещения, начала работы, повторного отсчёта времени (только при окончании уже запущенного процесса), завершения работы.



## 2.3 Создание полигональной сетки

В ходе работы программы будет активно использоваться полигональная сетка. Это совокупность вершин, рёбер и граней, которые определяют форму многогранного объекта в трёхмерной компьютерной графике и объёмном моделировании.

Гранями обычно являются треугольники, четырёхугольники или многоугольники, при этом у каждой грани высчитывается нормаль.

Наиболее удобно использовать треугольник, так как:

- 1) любой многоугольник можно разделить на несколько треугольников
- 2) треугольник также можно разбить на другие треугольники меньшей площади
- 3) треугольник не может быть невыпуклым, и это свойство упрощает работу
- 4) позволяют эффективно использовать память, так как можно организовать хранение так, чтобы для каждого следующего треугольника нужно будет добавлять только одну вершину

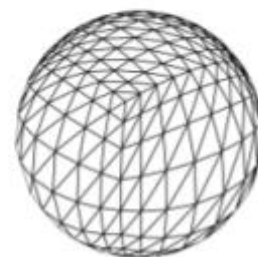
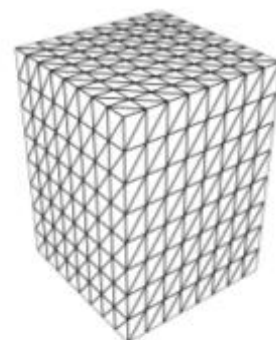


Рисунок 2 -  
Использование  
полигональной сетки

Примеры использования полигональной сетки приведены на Рисунке 2.

Построение полигональной сетки осуществляется следующим образом: зная число разбиений по осям, можно по рассматриваемой формуле плоскости найти неизвестное, тем самым заполнив “ячейку” сетки. Далее все полученные точки соединяются в четырёхугольники, и в каждом четырёхугольнике строится диагональ, которая разбивает его на два треугольника. Все параметры разбиения заданы таким образом, чтобы обеспечить не только качественную визуализацию, но и минимальные, насколько это возможно затраты по памяти.

## 2.4 Алгоритм z-буфера

Блок-схема этого алгоритма представлена ниже на Рисунке 3:

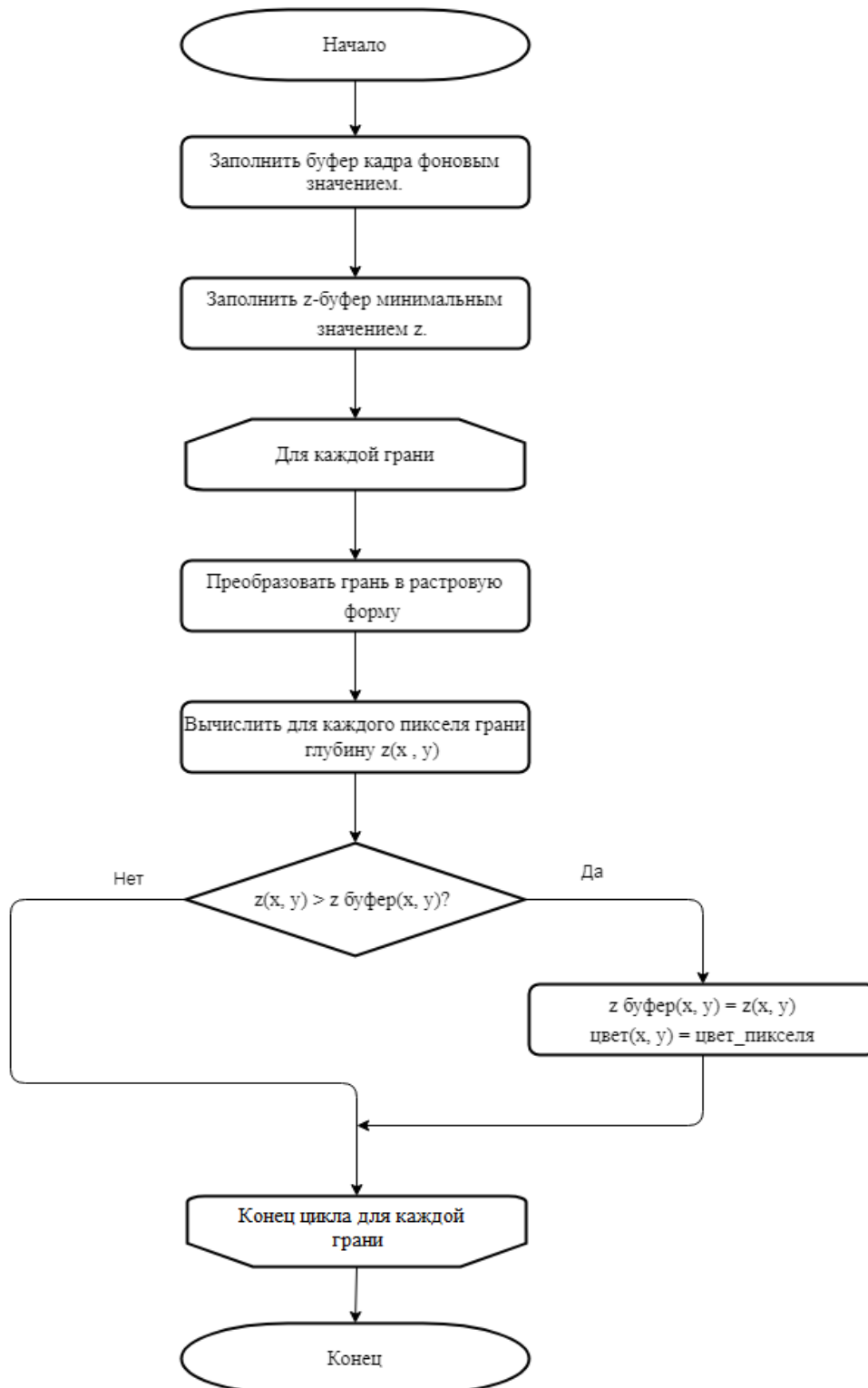


Рисунок 3 - Алгоритм z-буфера

Трудоёмкость этого алгоритма линейно зависит от числа поверхностей на сцене.

## 2.5 Алгоритм закрашки по Гуро

Сначала определяется интенсивность вершин, а затем вычисляется интенсивность соответствующего пикселя. Блок-схема представлена на Рисунке 4.

Этот метод хорошо сочетается с построчным сканированием. Для каждой сканирующей строки определяются её точки пересечения с рёбрами. В этих точках интенсивность вычисляется с помощью линейной интерполяции интенсивностей в вершинах рёбра. Затем для всех пикселей, находящихся внутри многоугольника и лежащих на сканирующей строке, аналогично вычисляется интенсивность.

Можно заранее высчитать нормали к каждой вершине, чтобы не делать эти вычисления в процессе работы алгоритма. Интенсивность в вершинах полигона определяется как скалярное произведение вектора светового луча и нормали в вершине.



Рисунок 4 - Алгоритм закрашки по Гуро

## 2.6 Модель освещения Фонга

Заданы: точечный источник света, поверхность, которая будет освещаться, сам наблюдатель. У каждой точки поверхности есть свои координаты, и определена нормаль к поверхности в этой точке.

Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения, и она делит угол между лучами на две равные части (Рисунок 5), где  $\vec{V}$  – направление на наблюдателя,  $\vec{L}$  – направление на источник,  $\vec{R}$  – отраженный луч.

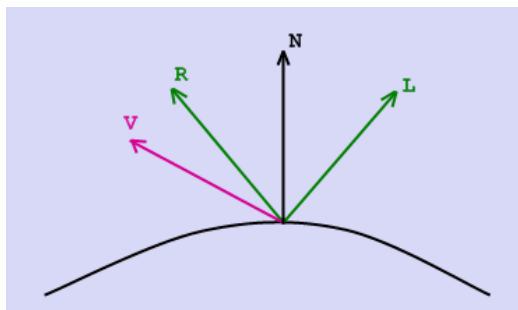


Рисунок 5 - Модель Фонга

Эта модель учитывает фоновую, рассеянную компоненты освещения и гляцевые блики.

По свойствам источника определяются мощности изучения каждой из трёх составляющих.

Для удобства все векторы считаются единичными, это необходимо для того, чтобы можно было сказать, что косинус угла между ними совпадает со скалярным произведением.

Интенсивность по модели Фонга рассчитывается следующим образом:

$$I = k_a I_a + k_d (\vec{N}, \vec{L}) + k_s (\vec{N}, \vec{V})^p, \text{ где}$$

$k_a$  – коэффициент фоновое освещение

$I_a$  – мощность фоновое освещение (заранее задаётся для всей сцены)

$k_d$  – коэффициент диффузного освещения

$\vec{N}$  – вектор нормали к поверхности в данной точке

$\vec{L}$  – направление из точки на источник

$k_s$  – коэффициент зеркального освещения

$\vec{V}$  – направление на наблюдателя

$p$  – коэффициент блеска

При фиксированном положении поверхности относительно источника освещения фоновая и рассеянные составляющие можно высчитать только один раз, так как они не зависят от положения наблюдателя. А зеркальная компонента зависит, поэтому её надо вычислять каждый раз, когда меняется положение наблюдателя.

## 2.7 Физика песка

Как упоминалось выше, песок будет разделён условно на три части: верхнюю, пересыпающуюся и нижнюю. Каждая из компонент подчиняется своим физическим законам, которые будут далее описаны. На Рисунке 6 показаны песочные часы в действии, можно различить все выделенные для работы части.



Рисунок 6 - Песочные часы

### Верхняя часть:

Условно разбивается на две составляющие: повторяющая форму часов боковая поверхность и образующая выемку в верхней части всего объёма. Обе компоненты задаются с помощью каркасной модели.

Боковая поверхность представляет собой перевернутую усечённую пирамиду с маленьким основанием в месте соприкосновения верхней и нижней части часов.

Поверхность, образующая выемку в песке, задаётся однородной полигональной сеткой, определяемой уравнением:

$$y(x, z) = - \frac{e^{-\frac{(mx^2 + mz^2)^2}{10}}}{k},$$

где коэффициенты  $m$ ,  $k$  зависят от времени.

С течением времени, когда глубина выемки достигнет своего максимума, вся система верхней части песка начнёт “проседать”, то есть уменьшаться в объёме. Другими словами, высота усечённой пирамиды будет равномерно уменьшаться, и поверхность с выемкой будет вместе с ней опускаться вниз. Так будет продолжаться до тех пор, пока всё содержимое часов не окажется в нижней части.

Экспериментально было выявлено, что на формирование выемки уходит примерно  $\frac{1}{10}t$ , где  $t$  – измерительное время, а  $\frac{9}{10}t$  на пересыпание.

Таким образом, за 1 единицу времени уровень песка будет снижаться на  $\frac{10h}{9t}$ , где  $h$  – начальный уровень. (!!!!!!!!!!!!!)

#### Нижняя часть:

Процессы и представление песка аналогичны тем, что были описаны в верхней части, только вместо выемки будет наблюдаться растущая со временем горка из песка, поверхность которой описывается следующим образом:

$$y(x, z) = \frac{e^{-\frac{(mx^2 + mz^2)^2}{10}}}{k},$$

где коэффициенты  $m$ ,  $k$  зависят от времени.

С течением времени, горка расти перестанет, но будет продолжать увеличиваться объём пересыпанного песка, и вся система нижней части будет равномерно подниматься (увеличивается высота усеченной пирамиды, которая описывает боковую поверхность уже нижней части песка, и поднимается поверхность с горкой). Процесс продолжается до тех пор, пока не пересыплется весь песок.

Время, которое затрачивается на каждый из процессов, совпадает с временными затратами, описанными в предыдущем пункте.

#### Пересыпающаяся часть:

Как было упомянуто выше, эта часть представляет собой совокупность частиц, которые имеют форму треугольных пирамид, описанные с помощью каркасной модели.

Все частицы будут падать с ускорением, и их скорость будет изменяться следующим образом:

$$v = at,$$

где  $v$  – скорость,  $a$  – ускорение,  $t$  – время.

И соответственно ордината будет меняться в соответствии с формулой:

$$y = y_0 - \frac{at^2}{2},$$

где  $y_0$ ,  $y$  – начальная и текущая ординаты.

## **2.8 Используемые типы и структуры данных**

В процессе реализации данного программного продукта используются следующие структуры данных:

1. Точка – хранит:

- 1.1. координаты  $x$ ,  $y$ ,  $z$

2. Вектор – хранит:
  - 2.1.Проекции на оси  $x, y, z$
3. Вершина – хранит:
  - 3.1.Точку
  - 3.2.Вектор нормали плоскости, проведённый из этой вершины
4. Ребро:
  - 4.1.Две Вершины
5. Грань:
  - 5.1.множество Вершин
  - 5.2.множество Рёбер
  - 5.3.Вектор нормали
  - 5.4.Цвет
6. Модель:
  - 6.1.множество Граней
  - 6.2.множество Вершин
7. Камера – хранит:
  - 7.1.Точку, определяющую положение
  - 7.2.Вектор, показывающий направление взгляда наблюдателя
8. Источник света:
  - 8.1.Точку, определяющую положение
  - 8.2.интенсивность
9. Ограничивающая поверхность (на которой находятся песочные часы)
  - 9.1. коэффициенты уравнения плоскости
10. Стекло – хранит:
  - 10.1. Модель
  - 10.2. коэффициенты прозрачности, отражения
11. Песок:
  - 11.1. Модель пересыпавшейся и не пересыпавшейся частей
  - 11.2. Модель песчинок



12. Песочные часы – хранит:

- 12.1.     измеряемое время
- 12.2.     общая высота
- 12.3.     Стекло
- 12.4.     Песок
- 12.5.     Модель

### **3.Технологический раздел**

#### **3.1 Выбор и обоснование языка программирования и среды разработки**

При выборе языка программирования важно учитывать много факторов в зависимости от задачи. В приложении на сцене будет большое количество объектов из-за строения самой модели. К тому же происходящее на сцене будет постоянно изменяться, что даёт дополнительную нагрузку.

В качестве языка программирования был выбран Си++. На это есть несколько причин:

1. при разработке этого программного продукта предусматривается использование объектно-ориентированного подхода, а его как раз поддерживает этот ЯП
2. этот язык обладает большой производительностью, что важно в данной задаче
3. предоставляет большое число шаблонов и библиотек, которые ускоряют работу и улучшают эффективность алгоритмов
4. накопившийся опыт работы с этим языком во время обучения, что упростит задачу, также в свободном доступе в большом количестве есть необходимая литература и документация

В качестве среды разработки был выбран QtCreator, который

1. хорошо знаком, так как активно использовался во время обучения
2. предоставляет удобную графическую библиотеку
3. легко работать с графическим интерфейсом
4. она бесплатна

#### **3.2 Формат входных данных**

Входным данным выступает измерительное время, которое пользователь выбирает из предлагаемого списка.

Пользователь также может задать через интерфейс расположение камеры, источника освещения.

### **3.3 Структура и состав классов**

### **3.4 Предоставляемый интерфейс**

Для работы пользователю предоставляется следующий интерфейс:

#### **4. Исследовательский раздел**

## **Заключение**

В процессе выполнения поставленной задачи была визуализирована трёхмерная сцена с множеством объектов. Были изучены, проанализированы и реализованы алгоритмы удаления невидимых линий и поверхностей, закраски, построения теней. Все алгоритмы выбирались, исходя из поставленной задачи. Была разработана физическая модель поведения объектов на сцене с учётом особенностей всей системы.

Итогом стал программный продукт, моделирующий работу песочных часов, отвечающий требованиям и предоставляющий пользователю заявленные возможности.

## Список использованной литературы

1. Роджерс Д. Алгоритмические основы машинной графики: Пер. с англ. – М.: Мир, 1989. – 512 с.
2. Методы представления дискретных трехмерных данных [Электронный ресурс]. – Режим доступа:  
[https://www.graphicon.ru/oldgr/ru/library/multires\\_rep/index.html](https://www.graphicon.ru/oldgr/ru/library/multires_rep/index.html) (дата обращения 02.07.20)
3. Алгоритмы закраски [Электронный ресурс]. – Режим доступа:  
[https://studbooks.net/2248060/informatika/odnotonnaya\\_zakraska\\_metod\\_graneniya](https://studbooks.net/2248060/informatika/odnotonnaya_zakraska_metod_graneniya) (дата обращения 03.07.20)
4. Обзор алгоритмов построения теней в реальном времени [Электронный ресурс]. – Режим доступа:  
<https://www.ixbt.com/video/realtimeshadows.shtml> (дата обращения 04.07.20)
5. Ошаровская Е.В., Солодка В.И. Синтез трехмерных объектов с помощью полигональных сеток, Цифровые технологии, 2012, No.12, 2-9
6. Простые модели освещения [Электронный ресурс]. – Режим доступа:  
<http://grafika.me/node/344> (дата обращения 05.07.20)
7. Продвинутое освещение. Модель Блинна-Фонга [Электронный ресурс]. – Режим доступа:  
<https://habr.com/ru/post/353054/> (дата обращения 06.07.20)
8. Куров А.В., Курс лекций по дисциплине “Компьютерная графика” [Текст]

## Приложение