МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
«Санкт-Петербургский политехнический университет Петра Великого»
Институт компьютерных наук и кибербезопасности

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01 Математика и компьютерные науки

Отчет о выполнении лабораторной работы №3 по дисциплине
«Алгоритмические основы компьютерной графики»
по теме:
«Визуализация физического процесса»

Студент,
группы 5130201/20101 _____ Тищенко А. А.

Преподаватель _____ Курочкин М. А.

«____»_____ 2025 г.

Санкт-Петербург, 2025

Содержание

1 Постановка задачи

Дано: Видеоролик, демонстрирующий физический процесс — течение воды из крана. На записи также видно формирование и падение нескольких капель с головки крана. Один из кадров данного процесса показан на рисунке 1.

Цель работы: программными средствами визуализировать наблюдаемый на видеозаписи процесс.

Для достижения цели требуется выполнить следующие шаги:

1. Изучить видеоматериал и выделить ключевые стадии динамики водного потока.

2. Разработать алгоритмы и подходы для визуальной реконструкции процесса.

3. Реализовать визуализацию с применением графических библиотек, обеспечив:

   (a) естественное отображение колебаний и искажений водной струи;

   (b) достоверную анимацию падения отдельных капель с учётом их прозрачности и размеров;

   (c) возможность параметрической настройки характеристик струи (амплитуда колебаний, скорость появления капель).

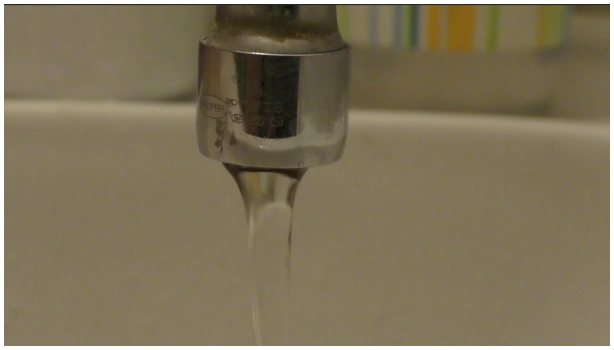

Рис. 1. Кадр видеофрагмента реального процесса

2 Описание физического процесса

Видео иллюстрирует физический процесс: течение воды из крана, который нобходимо было воспроизвести средствами процедурной анимации. Видео имеет разрешение 1600×913 пикселей, этого достаточно, чтобы заметить даже небольшие детали процесса, которые необходимо учесть при визуализации:

- **Динамика струи воды:**
  - В начальный момент из крана выходит непрерывный поток воды, формирующий вытянутую струю.
  - Струя не является идеально ровной — под действием колебаний и турбулентности она искажается, создавая небольшую рябь на границах струи.
  - Наблюдается постепенное движение воды влево и вправо, всего наблюдается 4 наиболее заметных и несколько небольших колебаний.
- **Образование капель:**
  - На видео наблюдается процесс формирования и падения двух капель.
  - Капли имеют разный размер и форму, а также формируются в разных местах и с разной скоростью на головке крана.
  - После формирования капли очень быстро падают вниз. Падение капель видно лишь на нескольких кадрах из-за небольшой частоты кадров видеозаписи.
- **Влияние фонового изображения:**
  - Вода частично прозрачна, поэтому за струёй виден фон.
  - Фоновые объекты, видимые через струю, сильно искажаются из-за преломления света в воде.

3 Общая структура визуализации

Фон и окружение: Для большего соответствия исходному видеоролику используется фоновое изображение, поверх которого накладывается визуализируемая струя воды. Это позволяет интегрировать анимацию в контекст сцены и повысить реализм изображения.

Струя воды: Основной элемент визуализации — поток воды из крана, основные положения которой формируются по контурам, заданным масками. Для каждого шага анимации контуры интерполируются, что обеспечивает плавное изменение формы струи во времени.

Дополнительные эффекты: Во время течения воды, появляются капли, которые отрываются от верхней границы струи и движутся вниз, усиливая динамику сцены.

3.1 Подход к формированию струи

Визуализация строится на геометрических контурах, которые задаются масками. Каждая маска определяет форму и положения струи в пространстве в определенный момент времени. Плавность перехода между масками обеспечивается интерполяцией контуров.

На форму добавляются синусоидальные колебания и случайные шумовые сдвиги, что позволяет создать характерную «живую» динамику воды.

Для придания глубины и текстуры поток заливается цветным градиентом, плавно переходящим от более тёмных тонов к светлым.

3.1.1 Маски формы струи

Каждая маска задаёт положение струи на определённой фазе. Пример такой маски приведен на рисунке 2 и рисунке 3. Последовательность масок определяет динамику движения струи влево и вправо.

3.1.2 Интерполяция между масками

При переходе от одной маски к другой вычисляются промежуточные контуры, которые подвергаются волновым и шумовым искажениям. Это позволяет струе двигаться и «колыхаться», имитируя естественное течение воды.

Рис. 2. Начальная маска положения струи

Рис. 3. Маска сдвинутого положения струи

3.2 Динамические эффекты

Искажения: Для отрисовки бликов и неоднородной структуры потока под саму струю была подложена картинка бликов, изображенная на рисунке 4. Для получения эффекта размытия на подложку струи накладывается горизонтальное синусоидальное смещение строк изображения, что создаёт эффект размытости и движения.

Градиентная заливка: Поток закрашивается набором полигонов с плавным изменением цвета и прозрачности, формируя характерное изменение цвета и прозрачности воды по времени ее стекания из крана.

Формирование капель: На верхнем крае струи периодически появляются капли. Капли визуализируются как вытянутые эллипсы с частичной прозрачностью.

Рис. 4. Подложка под струю, имитирующая блики и отражения окружающей среды

3.3 Алгоритм анимации и физические эффекты

- На каждом кадре выбираются начальный и конечный контур струи и вычисляется их интерполяция.
- К координатам контуров добавляются синусоидальные колебания и случайный шум.
- По получённым точкам строится маска, которая ограничивает область видимости подложки.
- Внутренняя область струи окрашивается с использованием цветового градиента.
- На верхнем крае струи (где она вытекает из под крана) периодически формируются капли. После форимрования капли падают вниз и скрываются за границей экрана.

Таким образом, реализованная структура визуализации позволяет имитировать динамику потока воды из крана с учётом геометрических деформаций, колебаний и отрыва капель, что делает анимацию достаточно реалистичной.

4 Процесс визуализации

Визуализация воспроизводит процесс течения воды из крана с последующим формированием струи и отдельных капель. Здесь применён геометрический подход: поток описывается через набор контуров, получаемых из масок и подвергаемых интерполяции и искажениям. Такая схема позволяет имитировать естественные колебания, разрывы и изменения формы водного потока.

4.1 Последовательность процесса визуализации

4.1.1 Формирование и динамика струи

- **Начальное появление:** Струя формируется на основе исходной маски (рисунок 2), которая задаёт начальную геометрию потока.

- **Изменение формы:** При переходе между масками используется интерполяция контуров. Дополнительно применяются синусоидальные колебания и шумовые искажения, что создаёт эффект подвижной струи с характерными колебаниями и неровностями.

- **Генерация капель:** На верхнем крае потока периодически появляются отдельные капли, которые движутся вниз под действием гравитации. Их размер и прозрачность варьируются, что делает анимацию более естественной.

4.1.2 Визуальные эффекты

- **Прозрачность и цвет:** Поток и капли визуализируются как полупрозрачные объекты с градиентной заливкой, что позволяет передать глубину, световые переходы и эффект преломления.

- **Фоновое изображение:** На задний план накладывается статическое фото, что усиливает реалистичность сцены и создаёт контекст окружающей среды.

- **Искажения:** Кадры подвергаются горизонтальному синусоидальному смещению, которое усиливает впечатление движения и дрожания струи.

4.1.3 Завершение анимации

- **Окончание симуляции:** Визуализация продолжается до достижения последней маски либо заданного числа кадров. Каждый кадр сохраняется для последующего формирования видеоряда.

4.2 Используемые технологии и библиотеки

Python — основной язык программирования.

4.2.1 Используемые библиотеки

- **Pygame** — для создания окна, загрузки изображений, построения анимации и работы с масками.
  - `pygame.init()` — инициализирует все основные модули Pygame.
  - `pygame.display.set_mode()` — создаёт окно приложения и поверхность для отрисовки.
  - `pygame.image.load()` — загружает изображения фона и масок.
  - `pygame.Surface()` — создаёт поверхности с поддержкой прозрачности, используемые для наложения эффектов.
  - `pygame.draw.polygon()` и `pygame.draw.ellipse()` — для отрисовки контура струи и капель воды.
  - `pygame.display.flip()` — обновляет окно, показывая отрисованный кадр.
  - `pygame.time.Clock()` — управляет частотой кадров анимации.
  - `pygame.event.get()` — отслеживает события (например, закрытие окна).
  - `pygame.quit()` — завершает работу программы и освобождает ресурсы.
- **NumPy** — для преобразования массива пикселей поверхности в формат, совместимый с записью видео.
- **Imageio** — для записи последовательности кадров анимации в файл `.mp4`.
- **math** — для вычисления синусоидальных искажений струи.

- **random** — для внесения случайных колебаний в контуры и траектории капель.

4.3 Основные числовые параметры

- *Размер окна:* определяется автоматически по загруженному изображению фона (например, 400×600 пикселей).
- *Частота кадров:* `FPS = 60` — обеспечивает плавность анимации.
- *Цвета струи:*
  - Начальный цвет: `(55, 50, 45, 100)` — тёмно-серый полупрозрачный.
  - Конечный цвет: `(180, 180, 180, 50)` — светло-серый с большей прозрачностью.
- *Анимация:*
  - `animation_speed = 0.005` — скорость перехода между фазами.
  - `amplitude = 2` — амплитуда волнового искажения контура.
  - `frequency = 0.05` — частота волны.
  - `t += 9` — параметр времени, задающий движение искажения.
- *Капли воды:*
  - Размер: `18` пикселей.
  - Скорость падения: `35` пикселей за кадр.
  - Цвет: полупрозрачный бело-голубой `(60, 55, 55, 30)`.
  - Частота появления: `drop_frequency = 1`.
- *Фазы анимации:*
  - Используются 4 маски (`mask_1.png` ... `mask_4.png`), определяющие форму струи на каждом этапе.
  - Интерполяция контуров плавно изменяет форму между фазами.
  - Переключение фаз происходит при завершении цикла интерполяции.

Таким образом, разработанная программа выполняет визуализацию движения водной струи с использованием масок, градиентной заливки и реалистичных капель, обеспечивая плавную анимацию и экспорт результата в видеофайл.

5 Результаты работы

Ниже на рисунках 5-10 приведено сравнение кадров из реального видеоролика и собственной реализации, в различные моменты времени.

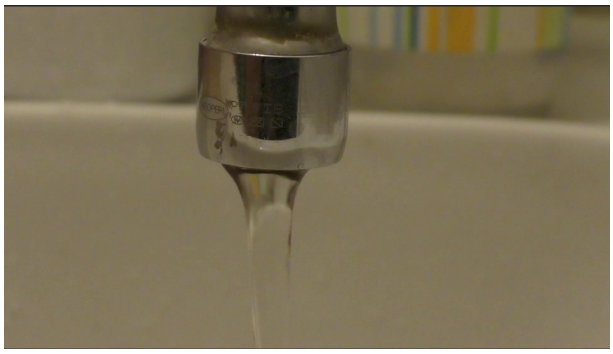

Рис. 5. Начальный момент оригинального видео

Рис. 6. Начальный момент реализации

Рис. 7. Серединный момент оригинального видео

Рис. 8. Серединный момент реализации

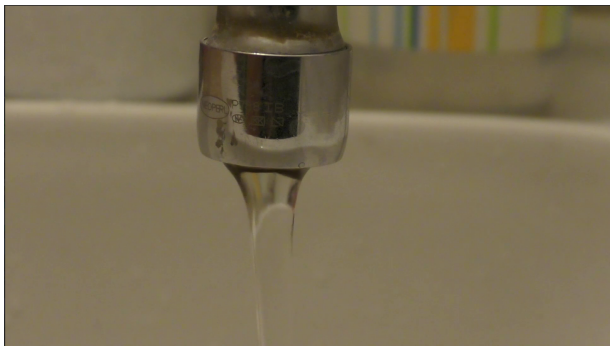

Рис. 9. Конечный момент оригинального видео

Рис. 10. Конечный момент реализации

Заключение

В ходе выполнения лабораторной работы был проведён анализ видеозаписи, отображающей физический процесс вытекания воды из крана и её дальнейшего движения. На основе изучения динамики струи были определены ключевые особенности: постепенное изменение формы потока, появление колебаний, формирование капель и влияние случайных искажений на контур.

Для воспроизведения наблюдаемого эффекта была реализована программная визуализация на языке Python. Визуализация основана на интерполяции масок, описывающих форму струи на различных этапах, а также на применении дополнительных эффектов, таких как волновое искажение, градиентная заливка и генерация падающих капель.

В технической части использованы следующие приёмы:

1. Применение альфа-канала для создания эффекта прозрачности воды;

2. Использование масок для задания формы струи и плавной смены фаз;

3. Добавление случайных искажений, формирующих реалистичные колебания потока;

4. Генерация отдельных капель, усиливающих правдоподобность анимации;

5. Экспорт последовательности кадров в видеофайл с помощью библиотеки `imageio`.

Разработанная визуализация позволяет достоверно отобразить процесс формирования и движения струи воды, включая её деформацию и каплеобразование. Программная реализация на базе библиотек `Pygame`, `NumPy`, `imageio`, `math` и `random` обеспечила гибкость настройки и высокую наглядность результата.

Созданная программа может быть использована как демонстрационный инструмент при изучении явлений гидродинамики, а также как основа для дальнейших экспериментов по визуализации жидкостей в компьютерной графике. Полный исходный код представлен в приложении А.

Приложение А. Код реализации программы

А.1 waterflow_visualization.py

Листинг 1. Визуализация течения воды из крана

```
import pygame
import math
import random
import imageio
import numpy as np

pygame.init()

# --- Класс для капель воды ---
class WaterDrop:
    def __init__(self, x, y):
        self.x = x
        self.y = y
        self.size = 18
        self.speed = 35

        # Прозрачный бело-голубой цвет (альфа 120)
        self.color = (60, 55, 55, 30)

        self.ellipse_width = self.size
        self.ellipse_height = self.size * 1.5

    def update(self):
        self.y += self.speed

    def draw(self, screen):
        drop_rect = pygame.Rect(
            self.x - self.ellipse_width / 2,
            self.y - self.ellipse_height / 2,
            self.ellipse_width,
            self.ellipse_height,
        )

        # Временная поверхность с альфой
        drop_surface = pygame.Surface(
        (self.ellipse_width, self.ellipse_height), pygame.SRCALPHA
        )

        # Рисуем каплю на этой поверхности
        pygame.draw.ellipse(drop_surface, self.color, (0, 0, self.
            ellipse_width, self.ellipse_height))

        # Накладываем на экран
        screen.blit(drop_surface, drop_rect.topleft)

    def is_offscreen(self, screen_height):
        return self.y > screen_height


```

```
# --- Настройка экрана и загрузка ресурсов ---
try:
    temp_background_image = pygame.image.load("waterflow_background.png")
    width, height = temp_background_image.get_size()
    use_background = True
except pygame.error as e:
    print(f"Error loading background image: {e}")
    width, height = 400, 600
    use_background = False

screen = pygame.display.set_mode((width, height))

if use_background:
    background_image = temp_background_image.convert()

clock = pygame.time.Clock()

masks = []
try:
    mask1_image = pygame.image.load("mask_1.png").convert_alpha()
    mask2_image = pygame.image.load("mask_2.png").convert_alpha()
    mask3_image = pygame.image.load("mask_3.png").convert_alpha()
    mask4_image = pygame.image.load("mask_4.png").convert_alpha()
    masks = [mask1_image, mask2_image, mask3_image, mask4_image]
except pygame.error as e:
    print(f"Error loading mask images: {e}")
    pygame.quit()
    exit()

# --- Вырезаем базовую подложку из отдельной картинки ---
if use_background:
    base_mask = pygame.image.load("moving_image_4.png").convert_alpha()
    stream_base = pygame.Surface((width, height), pygame.SRCALPHA)
    stream_base.blit(background_image, (0, 0))
    stream_base.blit(base_mask, (0, 0), special_flags=pygame.
        BLEND_RGBA_MULT)


# --- ФУНКЦИИ ---
def distort_only_stream(base, mask, t):
    """Размывает и искажает подложку по маске"""
    w, h = base.get_size()
    distorted = pygame.Surface((w, h), pygame.SRCALPHA)

    for y in range(0, h, 2):
        offset = int(5 * math.sin(0.05 * y + t * 0.1))  # сдвиг строки
        distorted.blit(base, (offset, y), (0, y, w, 2))

# применяем маску струи
    distorted.blit(mask, (0, 0), special_flags=pygame.BLEND_RGBA_MULT)
    return distorted


def get_stream_contours(mask_surface):
    left_contour = []
    right_contour = []
```

```
    rows_data = {}
    for y in range(mask_surface.get_height()):
        left_x = -1
        right_x = -1
        for x in range(mask_surface.get_width()):
            if mask_surface.get_at((x, y))[0] < 50:
                if left_x == -1:
                    left_x = x
                right_x = x
        if left_x != -1:
            rows_data[y] = (left_x, right_x)
    for y in sorted(rows_data.keys()):
            left_x, right_x = rows_data[y]
            left_contour.append((left_x, y))
            right_contour.append((right_x, y))
        return left_contour, right_contour


def make_stream_mask(left, right, width, height):
    mask_surface = pygame.Surface((width, height), pygame.SRCALPHA)
    polygon_points = left + right[::-1]
    pygame.draw.polygon(mask_surface, (255, 255, 255, 255),
        polygon_points)
    return mask_surface


def interpolate_countours(
start_left_contour_, start_right_contour_, end_left_contour_,
    end_right_contour_
):
    interpolated_left = []
    interpolated_right = []
    min_len_left = min(len(start_left_contour_), len(end_left_contour_))
    min_len_right = min(len(start_right_contour_), len(end_right_contour_
        ))

    for i in range(min_len_left):
        start_x, start_y = start_left_contour_[i]
        end_x, end_y = end_left_contour_[i]
        interp_x = start_x + (end_x - start_x) * animation_progress
        wave1 = amplitude * math.sin(frequency * start_y + t)
        wave2 = (amplitude / 2) * math.sin(2 * frequency * start_y + 1.5 *
            t)
        noise = random.uniform(-0.5, 0.5)
        x_offset = wave1 + wave2 + noise
        interpolated_left.append((interp_x + x_offset, start_y))

    for i in range(min_len_right):
        start_x, start_y = start_right_contour_[i]
        end_x, end_y = end_right_contour_[i]
        interp_x = start_x + (end_x - start_x) * animation_progress
        wave1 = amplitude * math.sin(frequency * start_y + t)
        wave2 = (amplitude / 2) * math.sin(2 * frequency * start_y + 1.5 *
            t)
        noise = random.uniform(-0.5, 0.5)
        x_offset = wave1 + wave2 + noise
```

```
155         interpolated_right.append((interp_x + x_offset, start_y))
156
157     return interpolated_left, interpolated_right
158
159
160 # --- Контуры масок ---
161 contours = []
162 for mask in masks:
163     left, right = get_stream_contours(mask)
164     contours.append([left, right])
165
166 # --- Параметры ---
167 start_color = (55, 50, 45, 100)
168 end_color = (180, 180, 180, 50)
169 animation_progress = 0
170 animation_speed = 0.005
171 amplitude = 2
172 frequency = 0.05
173 t = 0
174 reverse = False
175 phase = 1
176 drops = []
177 FPS = 60
178 drop_frequency = 1
179 drop_counter = 0
180
181 writer = imageio.get_writer("animation_with_blur.mp4", fps=FPS)
182
183 # --- Главный цикл ---
184 running = True
185 while running:
186     for event in pygame.event.get():
187         if event.type == pygame.QUIT:
188             running = False
189
190     # --- Отрисовка фона ---
191     if use_background:
192         screen.blit(background_image, (0, 0))
193     else:
194         screen.fill((255, 255, 255))
195
196     # Интерполяция контура струи
197     interpolated_left, interpolated_right = interpolate_countours(
198         contours[0][1], contours[0][0], contours[phase][1], contours[phase
            ][0]
199     )
200     stream_mask = make_stream_mask(interpolated_left, interpolated_right,
          width, height)
201
202     # --- Размытая и искажённая подложка ---
203     distorted_stream = distort_only_stream(stream_base, stream_mask, t)
204     screen.blit(distorted_stream, (0, 0))
205
206     # --- Цветная заливка для струи ---
207     polygon_points = interpolated_left + interpolated_right[::-1]
208     if interpolated_left and interpolated_right:
```

```
209         stream_surface = pygame.Surface((width, height), pygame.SRCALPHA)
210     num_points = len(interpolated_left)
211     for i in range(num_points - 1):
212         r = int(start_color[0] + (end_color[0] - start_color[0]) * (i /
                num_points) ** 0.6)
213         g = int(start_color[1] + (end_color[1] - start_color[1]) * (i /
                num_points) ** 0.6)
214         b = int(start_color[2] + (end_color[2] - start_color[2]) * (i /
                num_points) ** 0.6)
215         a = int(start_color[3] + (end_color[3] - start_color[3]) * (i /
                num_points) ** 0.2)
216         points = [
217             interpolated_left[i],
218             interpolated_left[i + 1],
219             interpolated_right[i + 1],
220             interpolated_right[i],
221         ]
222         pygame.draw.polygon(stream_surface, (r, g, b, a), points)
223     screen.blit(stream_surface, (0, 0))
224
225     coef = 3 if phase == 0 or phase == 1 else 1.5
226     # --- Анимация прогресса ---
227     animation_progress += animation_speed if not reverse else
            animation_speed * coef
228     if animation_progress >= 1.0 or animation_progress <= 0.0:
229         animation_speed *= -1
230         if animation_progress <= 0.0:
231             phase = min(3, phase + 1)
232             drop_counter +=1
233             reverse = not reverse
234     animation_progress = max(0.0, min(1.0, animation_progress))
235
236     t += 9
237     if drop_counter >= drop_frequency:
238         last_point_y = min(p[1] for p in polygon_points) + 25
239         bottom_points_x = [p[0] for p in polygon_points if p[1] ==
                last_point_y]
240     if bottom_points_x:
241         spawn_x = sum(bottom_points_x) / len(bottom_points_x)
242         drops.append(WaterDrop(spawn_x + random.randint(-5, 5),
                last_point_y))
243     drop_counter = 0
244
245     new_drops_list = []
246     for drop in drops:
247         drop.update()
248         drop.draw(screen)
249         if not drop.is_offscreen(height):
250             new_drops_list.append(drop)
251     drops = new_drops_list
252     pygame.display.flip()
253
254     # --- Запись кадра ---
255     frame = pygame.surfarray.array3d(screen)
256     frame = np.swapaxes(frame, 0, 1)
257     writer.append_data(frame)
```

```
258
259     clock.tick(FPS)
260
261 writer.close()
262 pygame.quit()
```