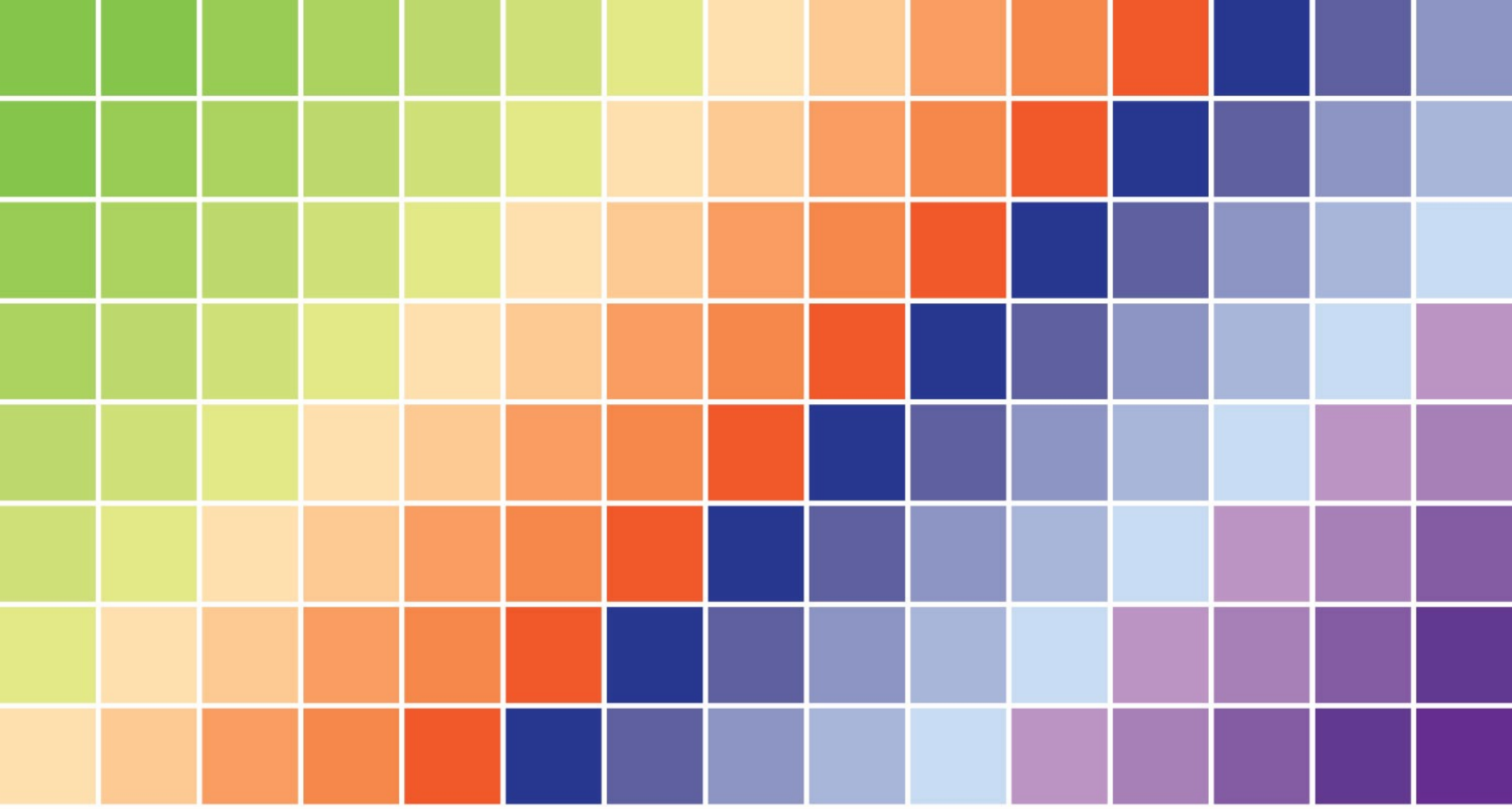

School of Computer Science

ЗВУК И АНИМАЦИЯ В ИГРАХ. РАЗРАБОТКА БОЛЬШОЙ ACTION-ИГРЫ

ПРОГРАММИРОВАНИЕ НА PYTHON

Лекции для IT-школы

РАЗРАБОТКА ACTION-ИГРЫ
ЧТО БУДЕМ ДЕЛАТЬ

- Читать клавиатурный ввод в графической игре
- Воспроизводить звук в программе
- Проигрывать музыкальные файлы
- Создавать анимацию
- Разрабатывать игровую программу, последовательно создавая все более и более сложные ее версии

ЗНАКОМСТВО С ИГРОЙ
ПРЕРВАННЫЙ ПОЛЕТ

- Пользователь управляет космическим кораблем, который окружают астероиды
- Космический корабль:
  - может вращаться и линейно ускоряться
  - вооружен ракетами и может их запускать
- Ракетный удар по астероиду разрушает его
- При разрушении астероида крупного и среднего размера они дробятся на части
- Как только игрок расправится со всеми астероидами, на корабль налетит новая, еще более мощная волна космических камней
- Счет игрока увеличивается с каждым разрушенным астероидом
- Как только звездолет столкнется с одним из них, игре наступит конец

ЧТЕНИЕ С КЛАВИАТУРЫ
SOUND & ANIMATION/READ_KEY.PY

- Нажимая разные клавиши, можно передвигать космический корабль
- Используем объект keyboard из модуля games
- Метод is_pressed() объекта keyboard возвращает True, если была нажата указанная клавиша
- В модуле games задан набор констант, представляющих разные клавиши

КЛАВИАТУРНЫЕ КОНСТАНТЫ
ПРАВИЛА ФОРМИРОВАНИЯ

- Каждое имя клавиатурной константы начинается с games.K_
- Для алфавитно-цифровой клавиши за префиксом должен следовать символ в нижнем регистре:
  - К примеру, константа, связанная с клавишей “A”, называется games.K_a
  - Константа, связанная с клавишей “1”, называется games.K_1
- Для остальных клавиш к префиксу добавляется их английское название прописными буквами:
  - Например, пробелу соответствует константа games.K_SPACE

ВРАЩЕНИЕ СПРАЙТА

SOUND & ANIMATION/ROTATE_SPRITE.PY

- Вращаем корпус космического корабля в зависимости от нажатой клавиши:
  - → по часовой стрелке
  - ← против часовой стрелки
  - 1 – вернуть в исходное положение с наклоном 0°
  - 2 – повернуться вправо на 90°
  - 3 – повернуться вверх ногами на 180°
  - 4 – совершить поворот на 270°
- Используем свойство angle у объекта-спрайта, которое задает угол его наклона в градусах от верхней точки

СОЗДАНИЕ АНИМАЦИИ
SOUND & ANIMATION/EXPLSION.PY

- **Анимация** – это последовательность картинок (кадров), которые отображаются одна за другой
- Используем класс Animation. Он производный от Sprite и наследует все его атрибуты
- Дополнительные свойства объекта класса Animation – n_repeats и repeat_interval
- Графические ресурсы для анимации:

ЗВУК И МУЗЫКА

SOUND & ANIMATION/SOUND_AND_MUSIC.PY

- **Звук** – объект, загруженный из WAV-файла
  - Формат WAV удобен для записи звука с микрофона
  - Звук сохраняется в переменную-объект вызовом games.load_sound() и проигрывается с помощью метода play() этого объекта
  - Звуки могут проигрываться одновременно по одному из 8-ми доступных каналов
- **Музыка** – проигрывание мелодии, загруженной из WAV, MP3, OGG или MIDI файла:
  - Доступен всего один канал для проигрывания
  - Используется один объект music из модуля games
  - Задействуются методы load(), play() и stop()

УТОЧНЕННАЯ ФУНКЦИОНАЛЬНОСТЬ ИГРЫ
ТРЕБОВАНИЯ

- Корабль должен вращаться и пробивать себе путь вперед, направляемый вводом с клавиатуры
- При нажатии определенной клавиши корабль должен выпускать ракеты
- Астероиды должны перемещаться по экрану с разными скоростями – как правило, маленькие быстрее больших
- Корабль, выпускаемые им ракеты и все астероиды должны, выходя за пределы графического экрана, огибать его, то есть появляться с противоположной стороны
- Если ракета поражает какой-либо другой объект на экране, то и она, и этот объект должны красиво взрываться

УТОЧНЕННАЯ ФУНКЦИОНАЛЬНОСТЬ ИГРЫ
ТРЕБОВАНИЯ. ПРОДОЛЖЕНИЕ

- Если корабль сталкивается с каким-либо другим объектом на экране, то и он, и этот объект должны красиво взрываться
- Когда корабль гибнет, игра прекращается
- При разрушении большого и среднего астероида должна появляться пара астероидов меньшего размера
- Каждый раз, когда игрок разрушает астероид, его счет должен увеличиваться
- Чем меньше астероид, тем больше очков начисляется за его уничтожение
- После уничтожения всех астероидов, на корабль должна налетать очередная, более массовая «волна» астероидов

КЛАССЫ ИГРЫ
ПРОЕКТИРОВАНИЕ «СВЕРХУ ВНИЗ»

- Ship(games.Sprite) – Космический корабль игрока
- Missile(games.Sprite) – ракета, выпускаемая кораблем
- Asteroid(games.Sprite) – астероид, летающий по экрану
- Explosion(games.Animation) – взрыв при столкновении движущихся по экрану спрайтов

МЕДИАРЕСУРСЫ ИГРЫ
ПРОЕКТИРОВАНИЕ «СВЕРХУ ВНИЗ»

- Картинка – изображение корабля (ship.bmp)
- Картинка – изображение ракеты (missile.bmp)
- По одной картинке для представления каждого из трех типов астероидов (asteroid_xxx.bmp)
- Набор изображений, представляющих взрыв (explosion1.bmp – explosion9.bmp)
- Звуковой файл для случая, когда корабль пробивает себе дорогу вперед (thrust.wav)
- Звуковой файл для запуска ракеты (missile.wav)
- Звуковой файл, сопровождающий взрыв (explosion.wav)
- Музыкальная тема (theme.mid)

СОЗДАНИЕ АСТЕРОИДОВ
ACTION GAME/ASTROCRASH01.PY

- Создаем класс Asteroid, унаследованный от спрайта
- Создаем астероиды разного размера – маленькие, средние и большие
- Маленькие астероиды имеют больший потенциал двигаться быстрее
- Реализуем логику огибания экрана в методе update()

ВРАЩЕНИЕ КОСМИЧЕСКОГО КОРАБЛЯ
ACTION GAME/ASTROCRASH02.PY

- Посередине экрана отображаем космический корабль, который можно вращать стрелками вправо-влево
- Добавляем класс Ship, унаследованный от спрайта
- Задаем минимальный угол вращения в константе Ship.ROTATION_STEP
- Реализуем логику вращения корабля в методе Ship.update()

ДВИЖЕНИЕ КОСМИЧЕСКОГО КОРАБЛЯ
ACTION GAME/ASTROCRASH03.PY

- Нажатием клавиши «стрелка вверх» включается двигатель и корабль летит в том направлении, в котором он ориентирован
- В метод Ship.update() добавляется обработка клавиши «стрелка вверх»:
  - Изменение скорости корабля зависит от константы Ship.VELOCITY_STEP
  - Компоненты скорости dx и dy рассчитываются по тригонометрическим формулам в зависимости от угла наклона корабля:
    - math.sin(angle) характеризует долю ускорения, которая должна быть передана горизонтальной скорости корабля
    - -math.cos(angle) – долю ускорения, сообщаемую вертикальной скорости корабля

СТРЕЛЬБА РАКЕТАМИ
ACTION GAME/ASTROCRASH04.PY

- При нажатии клавиши Пробел звездолет будет выпускать ракету
- Ракета вылетает в том направлении, в котором ориентирован корабль
- Пока-что ракеты безобидны, но в будущем они будут разрушать все, что встретят на своем пути
- Реализуем класс Missile(games.Sprite) и дорабатываем метод Ship.update()

УПРАВЛЕНИЕ ПЛОТНОСТЬЮ ОГНЯ
ACTION GAME/ASTROCRASH05.PY

- В 4-ой версии корабль мог выпускать до 50 ракет в секунду – это многовато
- Ограничиваем плотность огня – после выстрела на заданный период времени запрещаем пуск новых ракет
- Добавляем константу Ship.MISSILE_DELAY
- Создаем конструктор класса Ship, где определяем атрибут missile_wait
- Изменяем метод Ship.update():
  - Добавляем обратный отсчет missile_wait
  - Разрешаем пуск очередной ракеты, если missile_wait == 0
  - При пуске ракеты присваиваем Ship.MISSILE_DELAY в self.missile_wait

ОБРАБОТКА СТОЛКНОВЕНИЙ
ACTION GAME/ASTROCRASH06.PY

- В Missile.update() и Ship.update() добавлены проверки на столкновение с другими объектами
- В классы Asteroid, Ship и Missile добавлен метод die()
- Добавлена константа Asteroid.SPAWN
- В методе Asteroid.die() при уничтожении немаленького астероида порождаем два новых меньшего размера

ДОБАВЛЕНИЕ ВЗРЫВОВ
ACTION GAME/ASTROCRASH07.PY

- Выполнена реорганизация для объединения повторяющегося кода
- Класс Wrapper производный от games.Sprite – «огибатель» экрана
- Класс Collider производный от Wrapper – «погибатель», т.е. такой «огибатель», который гибнет при столкновениях
- Все рабочие классы наследуются от Wrapper и Collider
- Добавлен класс Explosion для генерации анимационных взрывов

УРОВНИ, ВЕДЕНИЕ СЧЕТА, МУЗЫКА
ACTION GAME/ASTROCRASH08.PY

- Добавлены несколько уровней
- Создан класс Game – новый класс, объект которого представляет игру как таковую. Методы Game:
  - play() – начало игры
  - advance() – переход на следующий уровень
  - end() – завершение игрового эпизода
- Объекты класса Asteroid и Ship ссылаются на объект класса Game, для вызова методов advance() и end()

ИДЕИ ДЛЯ РАЗВИТИЯ ПРОЕКТА

- Рассмотрите главу 12 в книге Майкла Доусона «Программируем на Python»
- Развивайте свой собственный проект

  или
- Опробуйте эти варианты развития игры «Прерванный полёт»:
  - Добавьте функцию торможения на кнопку «стрелка вниз» и функцию остановки на кнопку End
  - Кроме астероидов, добавьте в игру новый тип более злобных космических объектов, которые разбиваются на три части

СПАСИБО ЗА ВНИМАНИЕ !

ВОПРОСЫ ?

School of Computer Science