Второй отборочный этап

Индивидуальная часть

Трехмерное моделирование и анимация

Теоретические вопросы (часть 1)

Задача II.1.1.1. (1 балл)

Pole это вершина, имеющая

1. только нечетное количество ребер;
2. 3 или 5 ребер;
3. только четное количество ребер;
4. 6 ребер.

Ответ: 2; 4.

Задача II.1.1.2. (1 балл)

Pivot point это

1. точка, являющаяся центром поворота и масштабирования объекта;
2. точка, обозначающая начало действия анимации.

Ответ: 1.

Задача II.1.1.3. (1 балл)

Как называется элемент 3D объекта, с помощью которых строятся 3d модели, не полые внутри?

1. mesh;
2. poligon;
3. voxel.

Ответ: 3.

Задача II.1.1.4. (1 балл)

Вершины, ребра и грани все вместе образуют объект, называемый

1. surface;
2. mesh;
3. armature,

Ответ: 2.

Задача II.1.1.5. (1 балл)

Скелет, необходимый для управлением позой объекта называется

1. nurb-bones;
2. metaball;
3. rig.

Ответ: 3.

Задача II.1.1.6. (1 балл)

С помощью какого инструмента можно ограничить изменение объекта?

1. constraint;
2. transform locker;
3. custom transform.

Ответ: 1.

Задача II.1.1.7. (1 балл)

Базовые формы (куб, цилиндр) называются

1. particles;
2. shapes;
3. primitives;
4. pieces.

Ответ: 3.

Задача II.1.1.8. (1 балл)

Параметр «френель» используется для

1. создания отражений;
2. создания смещения вершин объекта;
3. расчета вектора поворота объекта.

Ответ: 1.

Задача II.1.1.9. (1 балл)

Текстуры, созданные с помощью алгоритмов, называются:

1. функциональные;
2. алгоритмические;
3. динамические;
4. процедурные.

Ответ: 4.

Задача II.1.1.10. (1 балл)

Нормаль это

1. вектор, сонаправленный с поверхностью;
2. вектор, перпендикулярный поверхности;
3. состояние души.

Ответ: 2.

Задача II.1.1.11. (1 балл)

Pinching — артефакт, возникающий из-за работы с subdivision surface с вершиной типа

1. isolated vertex;
2. sharp vertex;
3. pole.

Ответ: 3.

Задача II.1.1.12. (1 балл)

Создание и расчет физический явлений реального мира (сила притяжения, столкновение объектов) называется

1. dynamic;
2. real transform;
3. simulation.

Ответ: 3.

Задача II.1.1.13. (1 балл)

В UV-развертке отдельные части развертки, разделенные швами, называются

1. island;
2. piece;

3. real transform.

Ответ: 1.

Задача II.1.1.14. (1 балл)

Если у объекта A есть родительский объект B и этот объект B изменяют, будет ли изменяться объект A?

1. нет;
2. да.

Ответ: 2.

Задача II.1.1.15. (1 балл)

Если у объекта A есть родительский объект B и объект A изменяют, будет ли изменяться объект B?

1. да;
2. нет.

Ответ: 2.

Теоретические вопросы (часть 2)

Задача II.1.1.16. (1 балл)

Процесс объединения костей с объектом называется

1. animation;
2. rigging;
3. skinning.

Ответ: 3.

Задача II.1.1.17. (1 балл)

Если наложить части UV развертки друг на друга, то в этих местах

1. материал в этой части исчезнет;
2. возникнут оверлапы текстуры;
3. появятся дырки в меше.

Ответ: 2.

Задача II.1.1.18. (1 балл)

Вид спереди или сбоку, на котором отсутствуют перспективные искажения

1. неперспективный;
2. упрощенный;
3. ортогональный.

Ответ: 3.

Задача II.1.1.19. (1 балл)

Текстура, имитирующая освещение неровностей поверхности, называется

1. roughness map;
2. normal map;
3. specular map.

Ответ: 2.

Задача II.1.1.20. (1 балл)

Можно ли 3D-объектам давать нулевой размер?

1. да;
2. может быть;
3. нет.

Ответ: 1.

Задача II.1.1.21. (1 балл)

На UV-развертке оси координат называются

1. A, B;
2. U, V;
3. X, Y.

Ответ: 2.

Задача II.1.1.22. (1 балл)

Область экрана, на которой видно 3D-сцену, называется:

1. teleport;
2. viewport;
3. scene viewer;
4. gizmo.

Ответ: 2.

Задача II.1.1.23. (1 балл)

Является ли источник света мешем?

1. нет;
2. может быть;
3. да.

Ответ: 1.

Задача II.1.1.24. (1 балл)

Displacement карта работает только с нормалями?

1. да;
2. нет;
3. может быть.

Ответ: 2.

Задача II.1.1.25. (1 балл)

Можно ли редактировать нормали полигонов и вершин отдельно друг от друга:

1. нет;
2. иногда;
3. да.

Ответ: 3.

Задача II.1.1.26. (1 балл)

Пропорциональное редактирование можно использовать для редактирования:

1. все перечисленное;
2. частей одного объекта;
3. нескольких отдельных объектов.

Ответ: 1.

Задача II.1.1.27. (1 балл)

Backface culling применяется для:

1. ускорения рендера;
2. скрытия отображения внутренней части объекта;

3. оба варианта верны.

Ответ: 3.

Задача II.1.1.28. (1 балл)

Non-manifold геометрия это:

1. полигоны с перевернутыми нормалями;
2. расположенные внутри меша полигоны;
3. полигоны, расположенные друг над другом.

Ответ: 1; 2.

Задача II.1.1.29. (1 балл)

WXYZ режим вращения используется главным образом для:

1. вращения отдельных полигонов;
2. системы частиц объекта;
3. анимации объектов.

Ответ: 3.

Задача II.1.1.30. (1 балл)

Keyframes это:

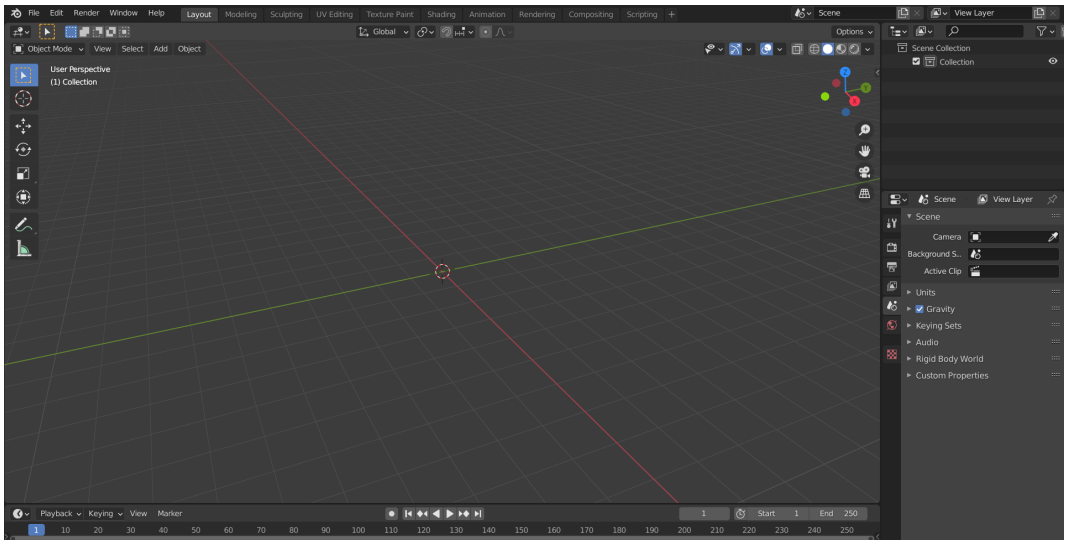
1. самые интересные кадры в фильме;
2. координаты камеры в режиме wireframes;
3. опорные точки меша;
4. ключевые кадры анимации.

Ответ: 4.

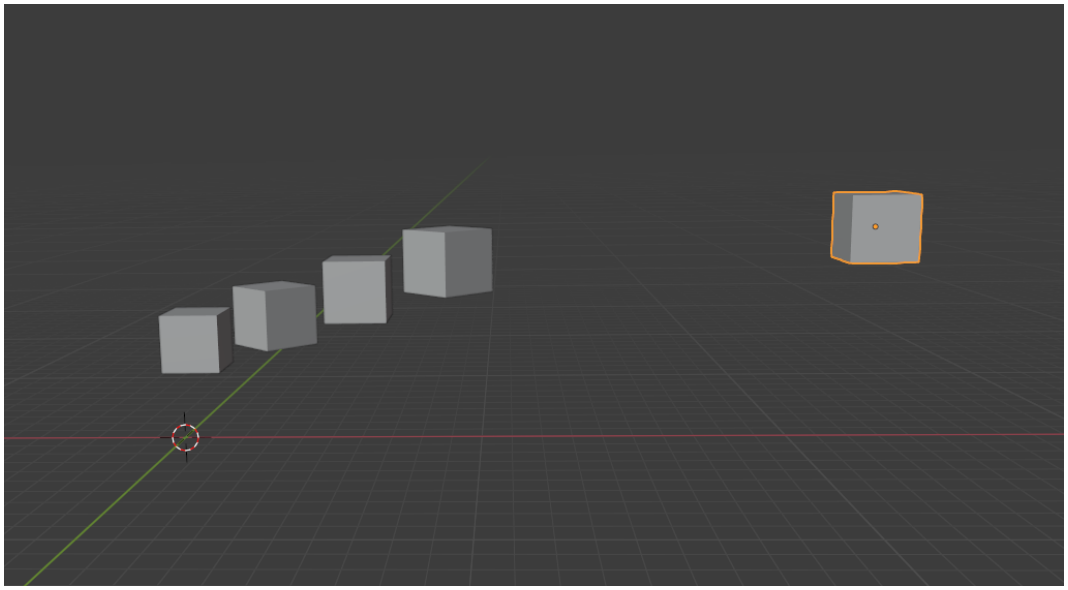
Практика со скриптами

Эта часть заданий содержит код, использующий библиотеку bpy для взаимодействия Python с Blender. По умолчанию считается, что она импортирована (import bpy уже есть в начале файла).

Расположите строки кода в таком порядке, чтобы в результате исполнения кода полученное изображение совпало с требуемым результатом. В каждом шаге прикреплено финальное изображение, которое получается при правильном расположении блоков кода. Все изображения предоставлены во фронтальной проекции. До выполнения скрипта сцена не содержит объектов (см. рисунок ниже)



Задача II.1.1.31. Расположите блоки кода в верном порядке (2 балла)



Расположите блоки кода в верном порядке для получения результата изображенного выше

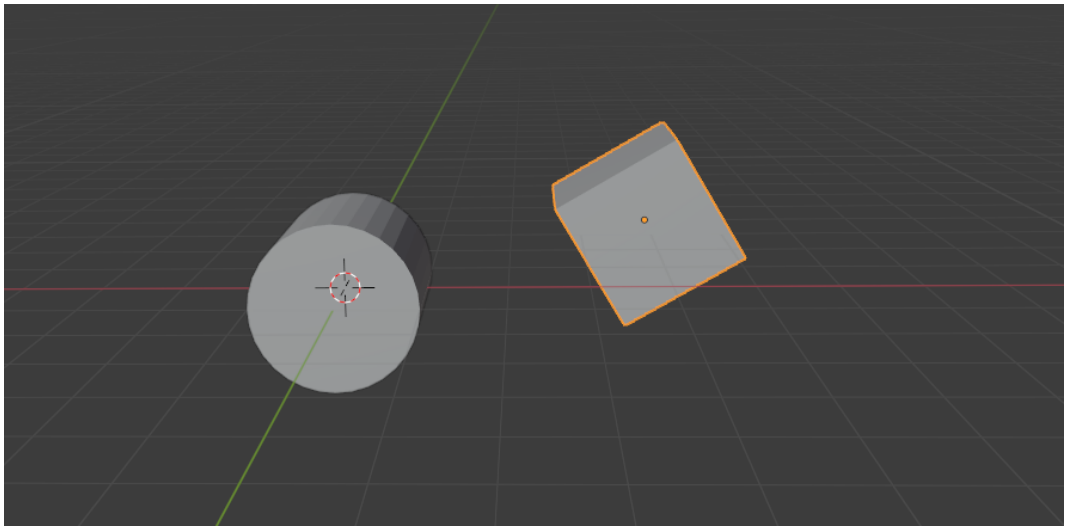
Номер блока	Код
1	object.location = [3 · i, 2 - i, 3 + i] object.rotation_euler[2] += (45 · 3.14 · i / 180)
2	for i in range(0,5):
3	bpy.ops.mesh.primitive_cube_add() object = bpy.context.active_object
4	object.location[0] += 10

В ответе напишите номера блоков кода расположенных в верном порядке

Например: 4231

Ответ: 2314.

Задача II.1.1.32. Расположите блоки кода в верном порядке (2 балла)



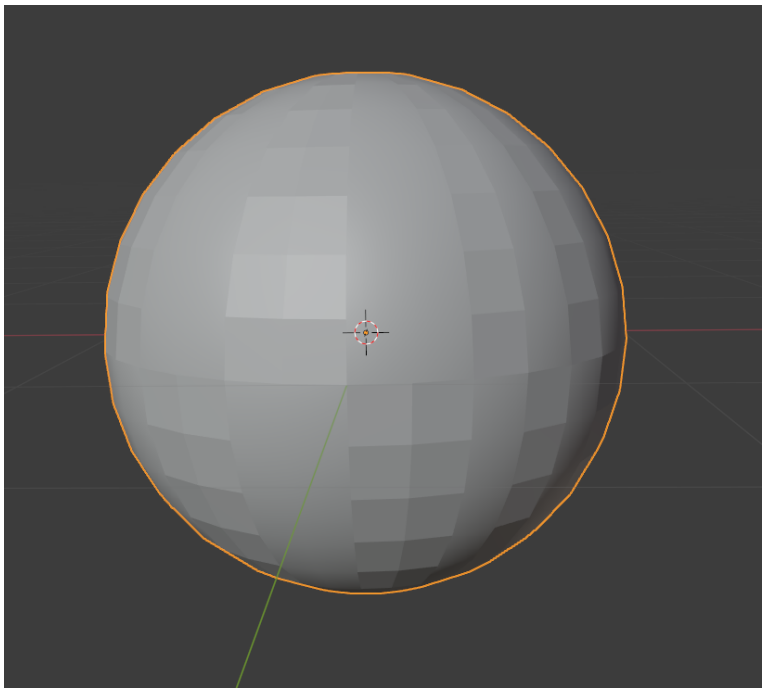
Расположите блоки кода в верном порядке для получения результата изображенного выше

Номер блока	Код
1	bpy.ops.mesh.primitive_cylinder_add() object = bpy.context.active_objec
2	object = bpy.context.active_object
3	object.rotation_euler[1] += (60 · 3.14 / 180) object.location = [4, 4, 0]
4	object.rotation_euler[0] += (90 · 3.14 / 180) bpy.ops.mesh.primitive_cube_add()

В ответе напишите номера блоков кода расположенных в верном порядке

Ответ: 1423.

Задача II.1.1.33. Расположите блоки кода в верном порядке (2 балла)



Расположите блоки кода в верном порядке для получения результата изображенного выше

Номер блока	Код
1	if polygon.index % 2 == 0:
2	bpy.ops.mesh.primitive_uv_sphere_add() mesh = bpy.context.active_object
3	polygon.use_smooth = True
4	for polygon in mesh.data.polygons:

В ответе напишите номера блоков кода расположенных в верном порядке

Ответ: 2413.

Задача II.1.1.34. Расположите блоки кода в верном порядке (2 балла)

Просмотрите видео фрагмент (`https://drive.google.com/file/d/1AvBLR5hk8qsnTkvZL7w0lJyNwn03-Vod/view?usp=sharing`), анимация в нем повторяется три раза для наглядности, с результатом того, что должно получится при верно составленном коде.

Расположите блоки кода в верном порядке для получения результата изображенного выше

Номер блока	Код
1	mesh.keyframe_insert(data_path ="location frame = 40) mesh.keyframe_insert(data_path ="rotation_euler frame = 10)
2	mesh.keyframe_insert(data_path ="location frame = 1)
3	mesh.keyframe_insert(data_path ="rotation_euler frame = 50)
4	bpy.context.scene.frame_end = 60 bpy.ops.mesh.primitive_uv_sphere_add() mesh = bpy.context.active_object
5	mesh.location = [4, 4, 0]
6	mesh.rotation_euler[2] += (99 · 3.14 / 180)

Ответом является комбинация из 6 чисел: номера блоков кода расположенных в верном порядке. Выберите верную комбинацию.

Ответ: 425163.

Задача II.1.1.35. Расположите блоки кода в верном порядке (2 балла)

Просмотрите видео фрагмент (`https://drive.google.com/file/d/1dUSbf0-S1FZPcl2o11efxTScMV_wvP5b/view?usp=sharing`), анимация в нем повторяется три раза для наглядности, с результатом того, что должно получится при верно составленном коде.

Расположите блоки кода в верном порядке для получения результата изображенного выше

Номер блока	Код
1	mesh.scale = [0.2, 0.2, 0.2] mesh.location = [i, 0, 5]
2	bpy.context.scene.frame_end = 60 for i in range(0,6):
3	bpy.ops.mesh.primitive_uv_sphere_add() mesh = bpy.context.active_object
4	mesh.keyframe_insert(data_path ="location frame = 40 - (i · 2))
5	mesh.keyframe_insert(data_path ="location frame = i · 2)
6	mesh.location = [i, 0, 0]

Ответом является комбинация из 6 чисел: номера блоков кода расположенных в верном порядке. Выберите верную комбинацию.

Ответ: 231564.

Геометрия и пространственное мышление

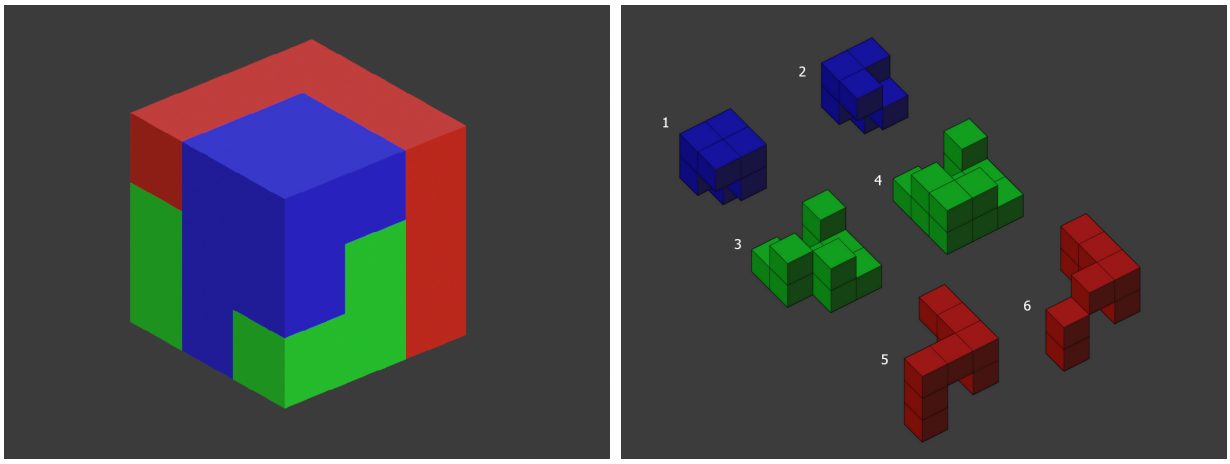
Предстоящие несколько задач призваны проверить ваше пространственное мышление.

Мы продублировали задание на каждом шаге, хотя условия одинаковые в каждом случае.

Задача II.1.1.36. Найти части, из которых состоит фигура (1 балл)

Вам дано изображение составной фигуры. Ваша задача — найти части, из которых состоит фигура (некоторые части лишние).

Фигуры пронумерованы слева направо сверху вниз (для первой шестерки фигур 1 — синяя фигура слева, 2 — синяя фигура справа, 3 — зеленая фигура слева, 4 — зеленая фигура справа, 5 — красная фигура слева, 6 — красная фигура справа). Номера дополнительно указаны на иллюстрации.

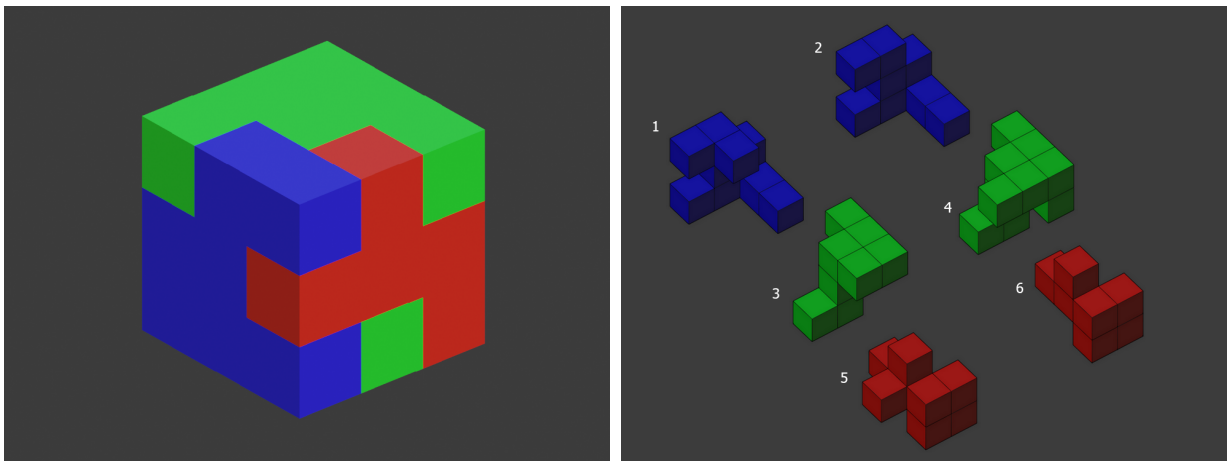


Ответ: 1; 3; 5.

Задача II.1.1.37. Найти части, из которых состоит фигура (1 балл)

Вам дано изображение составной фигуры. Ваша задача — найти части, из которых состоит фигура (некоторые части лишние).

Фигуры пронумерованы слева направо сверху вниз (для первой шестерки фигур 1 — синяя фигура слева, 2 — синяя фигура справа, 3 — зеленая фигура слева, 4 — зеленая фигура справа, 5 — красная фигура слева, 6 — красная фигура справа). Номера дополнительно указаны на иллюстрации.

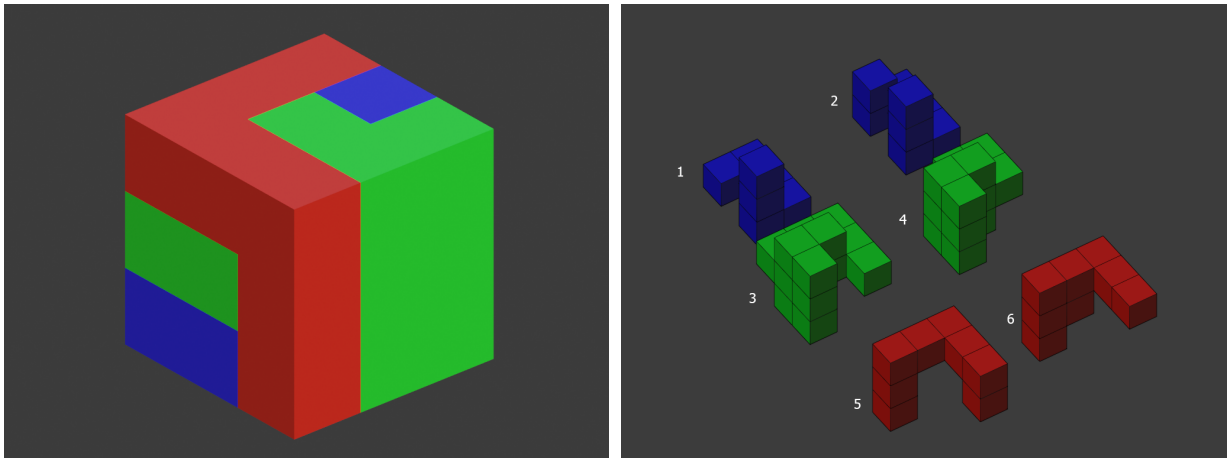


Ответ: 2; 4; 6.

Задача II.1.1.38. Найти части, из которых состоит фигура (1 балл)

Вам дано изображение составной фигуры. Ваша задача — найти части, из которых состоит фигура (некоторые части лишние).

Фигуры пронумерованы слева направо сверху вниз (для первой шестерки фигур 1 — синяя фигура слева, 2 — синяя фигура справа, 3 — зеленая фигура слева, 4 — зеленая фигура справа, 5 — красная фигура слева, 6 — красная фигура справа). Номера дополнительно указаны на иллюстрации.

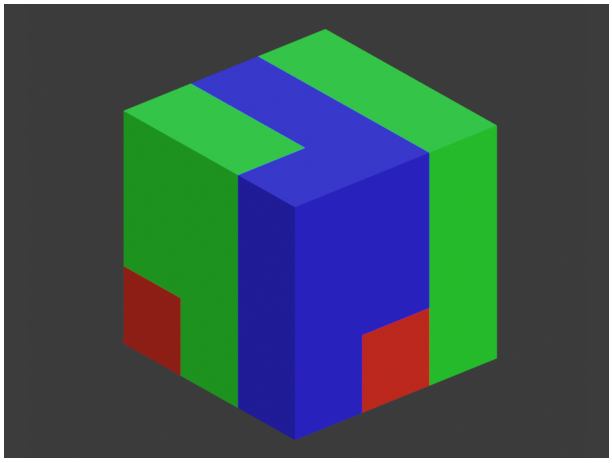


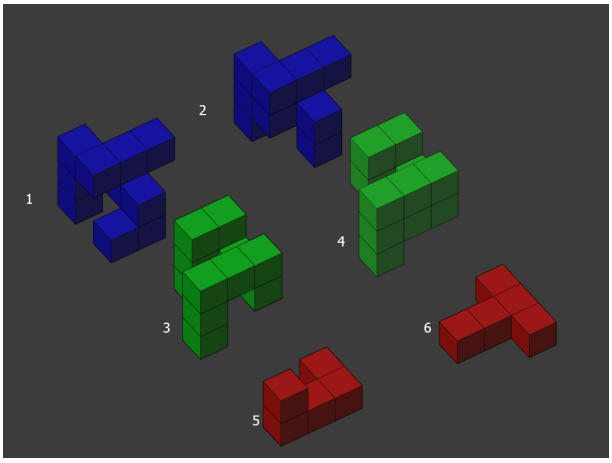
Ответ: 1; 4; 5.

Задача II.1.1.39. Найти части, из которых состоит фигура (1 балл)

Вам дано изображение составной фигуры. Ваша задача — найти части, из которых состоит фигура (некоторые части лишние).

Фигуры пронумерованы слева направо сверху вниз (для первой шестерки фигур 1 — синяя фигура слева, 2 — синяя фигура справа, 3 — зеленая фигура слева, 4 — зеленая фигура справа, 5 — красная фигура слева, 6 — красная фигура справа). Номера дополнительно указаны на иллюстрации.



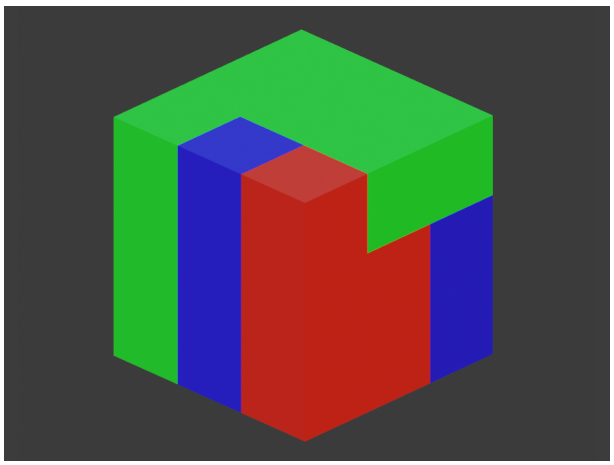


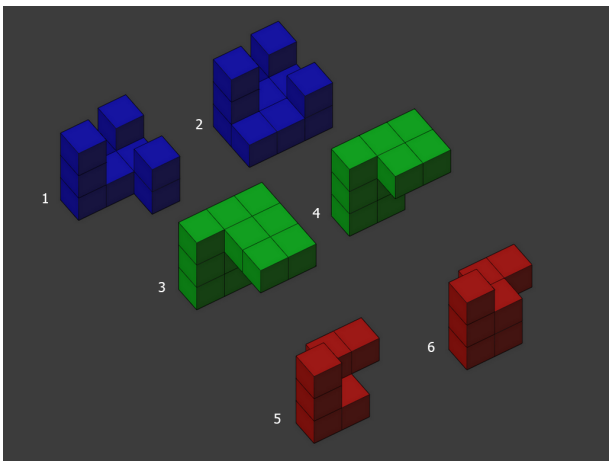
Ответ: 2; 3; 6.

Задача II.1.1.40. Найти части, из которых состоит фигура (1 балл)

Вам дано изображение составной фигуры. Ваша задача — найти части, из которых состоит фигура (некоторые части лишние).

Фигуры пронумерованы слева направо сверху вниз (для первой шестерки фигур 1 — синяя фигура слева, 2 — синяя фигура справа, 3 — зеленая фигура слева, 4 — зеленая фигура справа, 5 — красная фигура слева, 6 — красная фигура справа). Номера дополнительно указаны на иллюстрации.



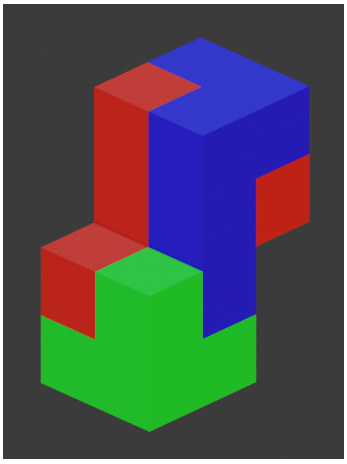


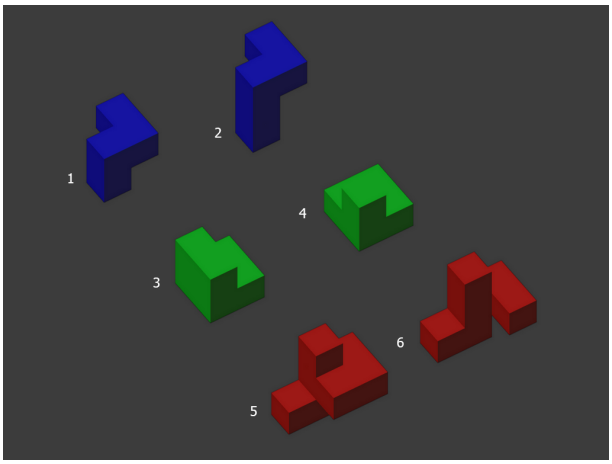
Ответ: 1; 3; 6.

Задача II.1.1.41. Найти части, из которых состоит фигура (1 балл)

Вам дано изображение составной фигуры. Ваша задача — найти части, из которых состоит фигура (некоторые части лишние).

Фигуры пронумерованы слева направо сверху вниз (для первой шестерки фигур 1 — синяя фигура слева, 2 — синяя фигура справа, 3 — зеленая фигура слева, 4 — зеленая фигура справа, 5 — красная фигура слева, 6 — красная фигура справа). Номера дополнительно указаны на иллюстрации.



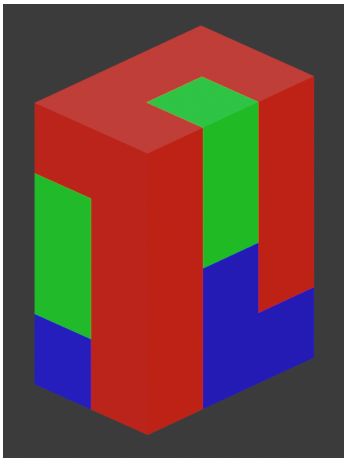


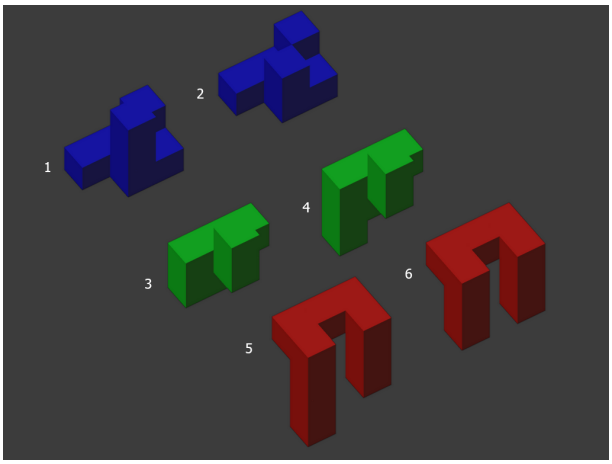
Ответ: 2; 4; 6.

Задача II.1.1.42. Найти части, из которых состоит фигура (1 балл)

Вам дано изображение составной фигуры. Ваша задача — найти части, из которых состоит фигура (некоторые части лишние).

Фигуры пронумерованы слева направо сверху вниз (для первой шестерки фигур 1 — синяя фигура слева, 2 — синяя фигура справа, 3 — зеленая фигура слева, 4 — зеленая фигура справа, 5 — красная фигура слева, 6 — красная фигура справа). Номера дополнительно указаны на иллюстрации.



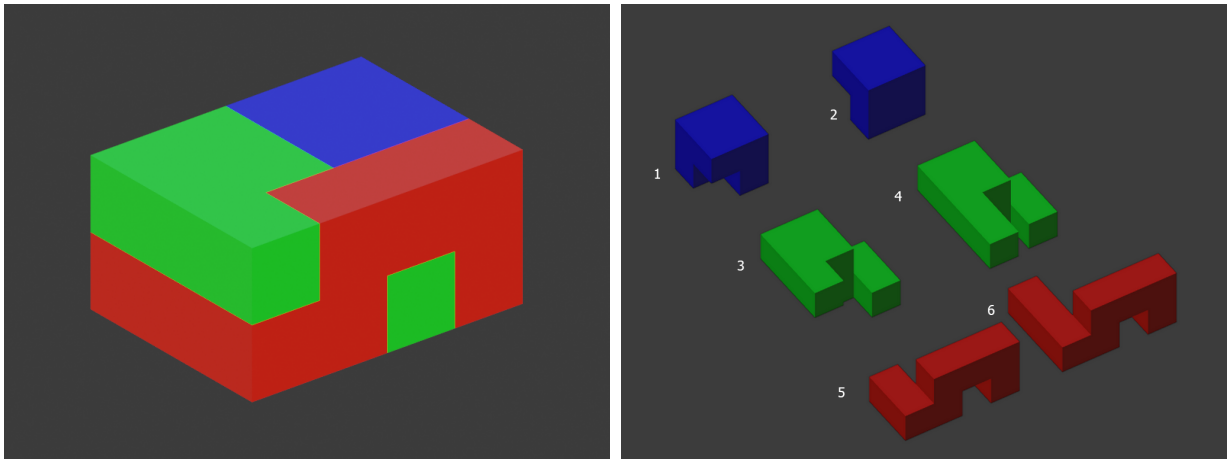


Ответ: 2; 3; 5.

Задача II.1.1.43. Найти части, из которых состоит фигура (1 балл)

Вам дано изображение составной фигуры. Ваша задача — найти части, из которых состоит фигура (некоторые части лишние).

Фигуры пронумерованы слева направо сверху вниз (для первой шестерки фигур 1 — синяя фигура слева, 2 — синяя фигура справа, 3 — зеленая фигура слева, 4 — зеленая фигура справа, 5 — красная фигура слева, 6 — красная фигура справа). Номера дополнительно указаны на иллюстрации.

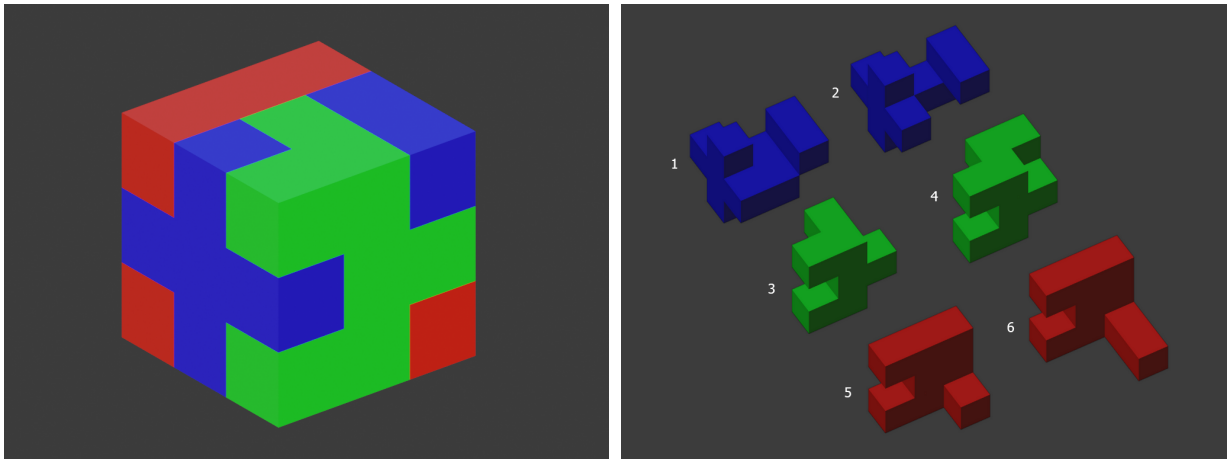


Ответ: 1; 3; 6.

Задача II.1.1.44. Найти части, из которых состоит фигура (1 балл)

Вам дано изображение составной фигуры. Ваша задача — найти части, из которых состоит фигура (некоторые части лишние).

Фигуры пронумерованы слева направо сверху вниз (для первой шестерки фигур 1 — синяя фигура слева, 2 — синяя фигура справа, 3 — зеленая фигура слева, 4 — зеленая фигура справа, 5 — красная фигура слева, 6 — красная фигура справа). Номера дополнительно указаны на иллюстрации.

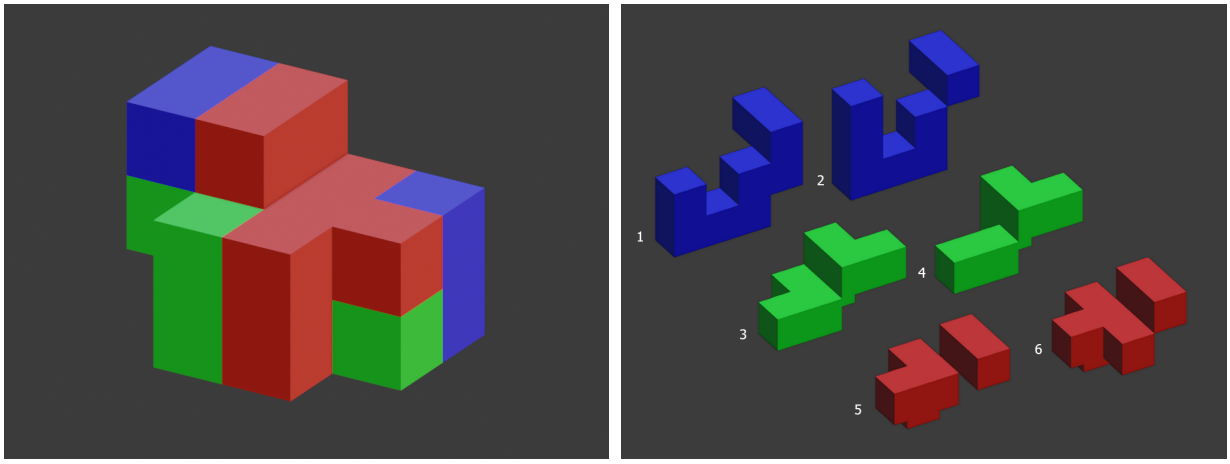


Ответ: 2; 3; 6.

Задача II.1.1.45. Найти части, из которых состоит фигура (1 балл)

Вам дано изображение составной фигуры. Ваша задача — найти части, из которых состоит фигура (некоторые части лишние).

Фигуры пронумерованы слева направо сверху вниз (для первой шестерки фигур 1 — синяя фигура слева, 2 — синяя фигура справа, 3 — зеленая фигура слева — , 4 — зеленая фигура справа, 5 — — красная фигура слева, 6 — красная фигура справа). Номера дополнительно указаны на иллюстрации.



Ответ: 1; 4; 6.

Разработка приложений дополненной реальности. AR-инфографика

Введение

В третьем модуле расположены задачи по дополненной реальности, связанные со сборкой приложений. Для выполнения заданий вам потребуется игровой движок Unity и Vuforia SDK. Верное решение любой задачи этого модуля, позволит вам реализовать приложение дополненной реальности, использование которого, в свою очередь, является ключом к ответу на платформе Stepik.

Для корректной работы, пожалуйста, используйте версию Unity 2019.4. Все проекты собраны под данную версию. При конвертации проектов в другие версии могут возникнуть проблемы с совместимостью частей решения, предложенных разработчиками данного модуля. В каждом задании есть ссылка на архив с проектом, в который уже импортирована Vuforia.

Если вы до этого разрабатывали в Unity и не хотите устанавливать новую версию, вы можете оставаться в вашей рабочей версии, но тогда вам необходимо перенести содержимое наших заготовленных проектов, в проекты созданные в вашей версии.

Если у вас возникают проблемы: в Unity не находится Vuforia или не получается настроить ее конфигурацию, ниже можно найти архивы всех проектов без включения библиотеки Vuforia.

Важные ссылки

- Ссылка на все проекты с Vuforia: https://drive.google.com/drive/folders/15dkDLKFqqAJ81qAfK16czbi9Pz60L-qQ?usp=sharing
- Ссылка на все проекты без Vuforia: https://drive.google.com/drive/folders/1PC-O6291xzDQ_blaGGeSXnESISAxVyYD?usp=sharing
- Ссылка на все маркеры(на всякий случай): https://drive.google.com/drive/folders/14AdIwIPYcVvXG25iQDiqqIM16UhPDe4j?usp=sharing

Обратите внимание, что для создания рабочего приложения маркерной дополненной реальности с использованием библиотеки Vuforia, вам необходимо импортировать пакет с маркером и лицензионный ключ. Без этих компонентов, у вас просто не будет возможности воспользоваться функциями библиотеки.

Чтобы в проектах не выходило такого, что маркеры не сходятся по размеру с оверлеем, всегда:

- печатайте изображения маркеров на бумаге размера А4;
- при создании маркеров на сайте устанавливайте ширину равную 1.

Задача II.1.2.1. Ключевая фраза (5 баллов)

Олимпиада КД НТИ — это не только соревнование, но и образовательная программа. Целью ее является не оценка, а обучение. Наше первое задание — познакомит вас с технологиями дополненной реальности. Мы даем к его выполнению наиболее полную инструкцию.

Ваша задача состоит в том, чтобы собрать проект, где на маркере написана пер-

вая часть фразы, а в оверлее будет вторая часть. Изображение маркера мы вам предоставляем — его можно найти в корне проекта. Оверлей вам надо собрать самим.

Для сборки оверлея создайте пустой объект и перетащите скрипт на этот пустой объект. У вас в объекте появился компонент Create Overlay. У этого компонента есть поля: Prefab, X, Y, Step. Для данного компонента вам надо перетащить префаб из папки /Assets/Prefabs в поле Prefab.

Чтобы получить верный ответ вам надо не только собрать и запустить проект, но и ориентируясь на размер оверлея, подобрать подходящие параметры для скриптов и префаба. Для изменения параметров префаба кликните на него 2 раза. Для изменения параметров скрипта не надо открывать и менять скрипт, вместо этого откройте тот самый пустой объект, на который вы перетаскивали скрипт и меняйте параметры компонента Create Overlay.

Проект-заготовка состоит из:

- скрипт для генерации продолжения фразы, /Assets/Scripts/CreateOverlay.cs
- один префаб — куб и его материал (с помощью префаба происходит генерация фразы), /Assets/Prefabs
- картинка, /Assets/A4_task1.jpg

Заготовка для проекта — `https://drive.google.com/file/d/1mfFu-dnoAAvrDwGg7PqqZoCG8yxjiHwq/view?usp=sharing`

Запустите приложение. Какая фраза всплывает перед глазами, после наведения камеры на маркер?

Выберите один вариант из списка

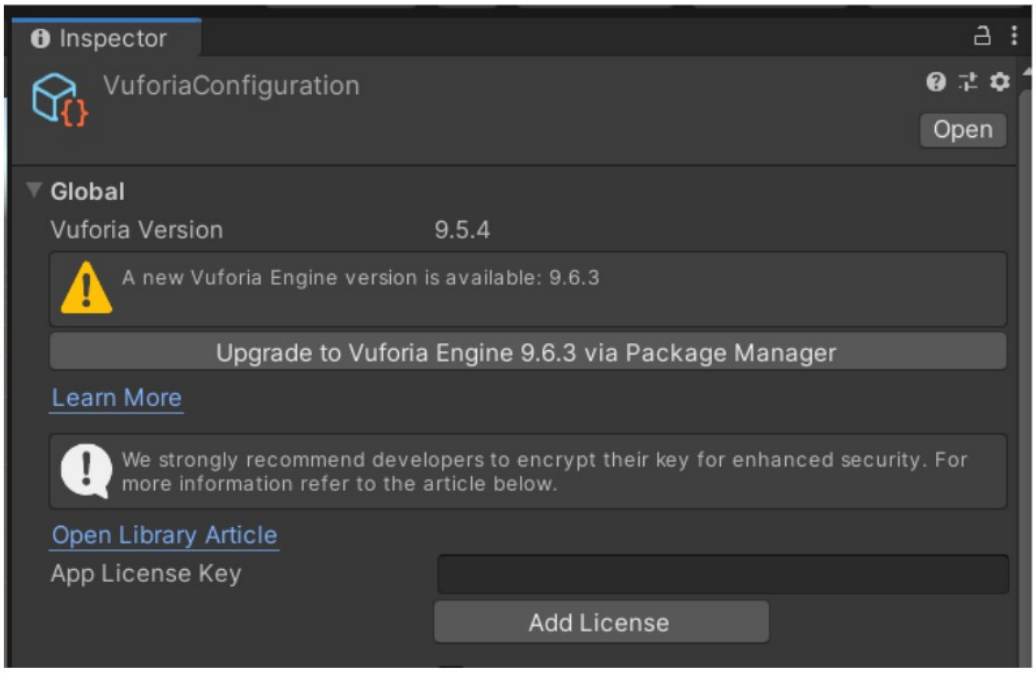
1. AR-world is our AltFuture as AltFuture is our duty.
2. AR-world is our AltFuture as AltFuture is notre chore.
3. AR-world is our AltFuture as AltFuture is notre deuoir.
4. AR-world is our AltFuture as AltFuture is notre beuoir.
5. AR-world is our AltFuture as AltFuture is our chore.
6. AR-world is our AltFuture as AltFuture is notre devoir.

Решение

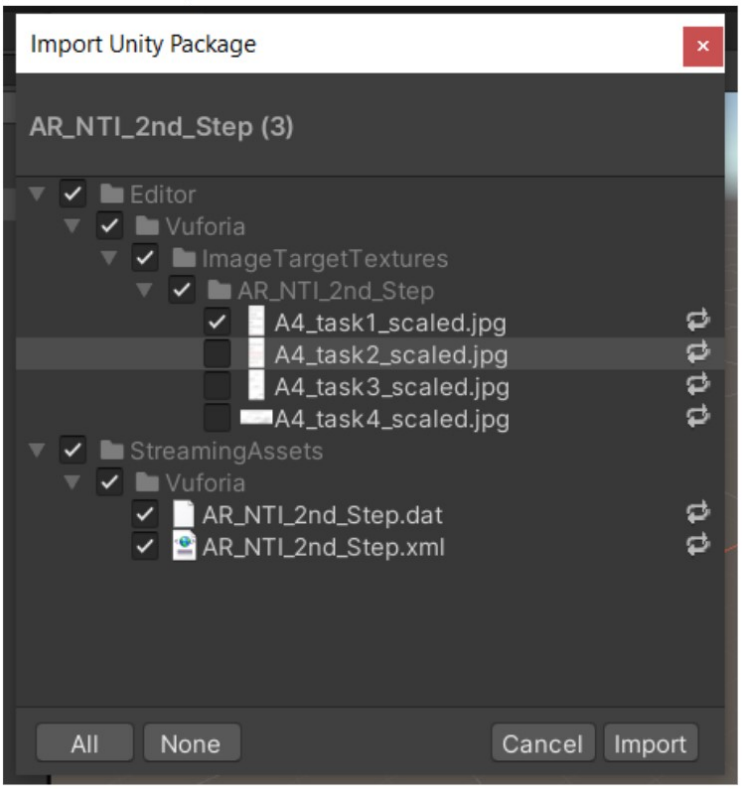
Как уже было сказано в условии, чтобы получить верный ответ вам надо запустить заготовку проекта, но и ориентируясь на размер оверлея, подобрать подходящие параметры для скриптов и префаба.

Для начала необходимо подключить вуфорию. Для этого заходим на главном меню Window -> Package Manager, в открывшемся окне выбираем Vuforia Engine.

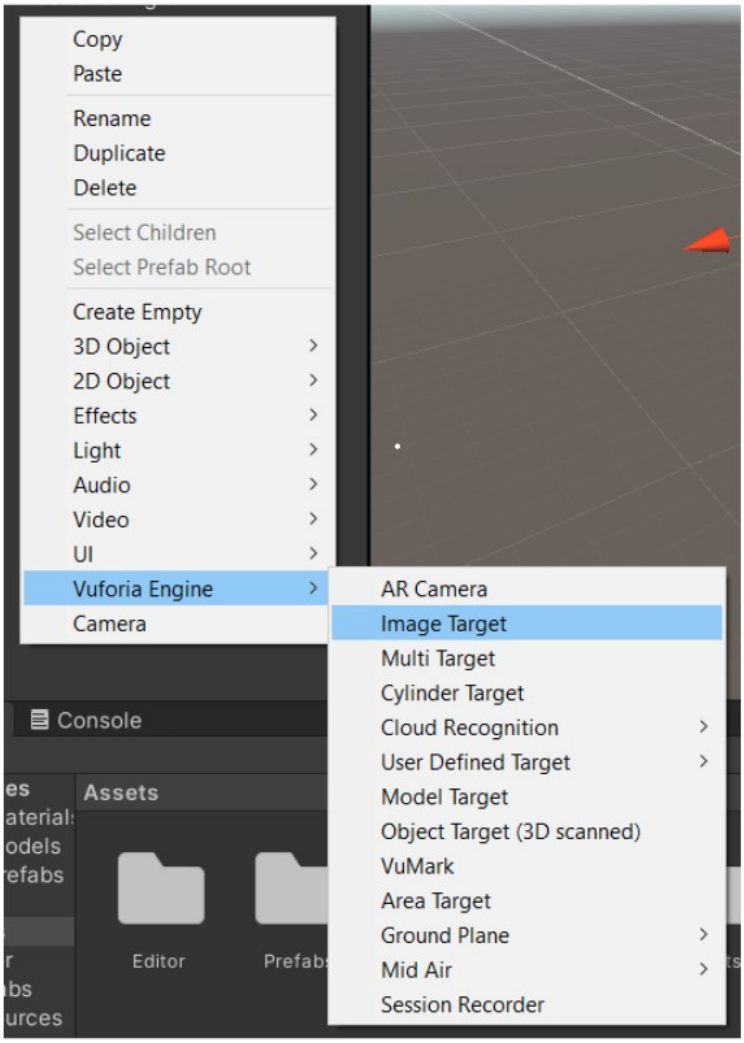
Затем нам необходимо настроить вухорию: идем Window -> Vuforia Configuration. В открывшемся окне инспектора необходимо заполнить поле App License Key ключом который вы сгенерируете на сайте (Vuforia developer portal).



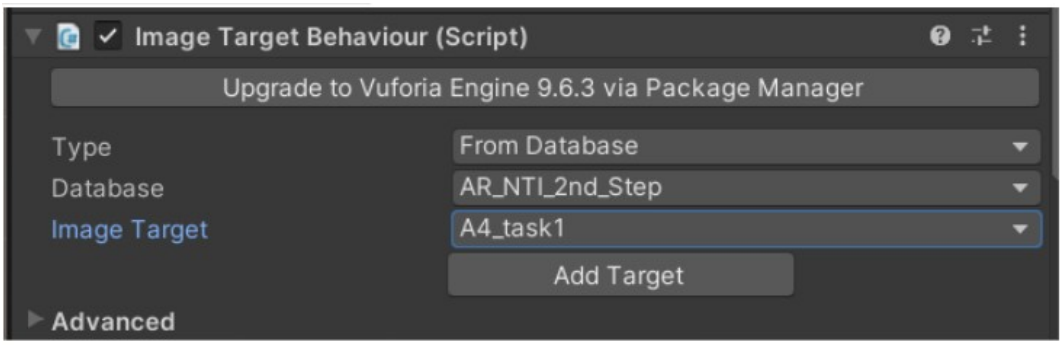
После этого необходимо подключить датасет с предобработаными картинками, который вам так же необходимо скачать с сайта. Чтобы загрузить датасет перейдите Assets -> Import Package -> Custom Package. После выбора нужного файла у вас откроется следующее окно, где вы можете выбрать какие из маркеров вы хотите использовать в проекте.



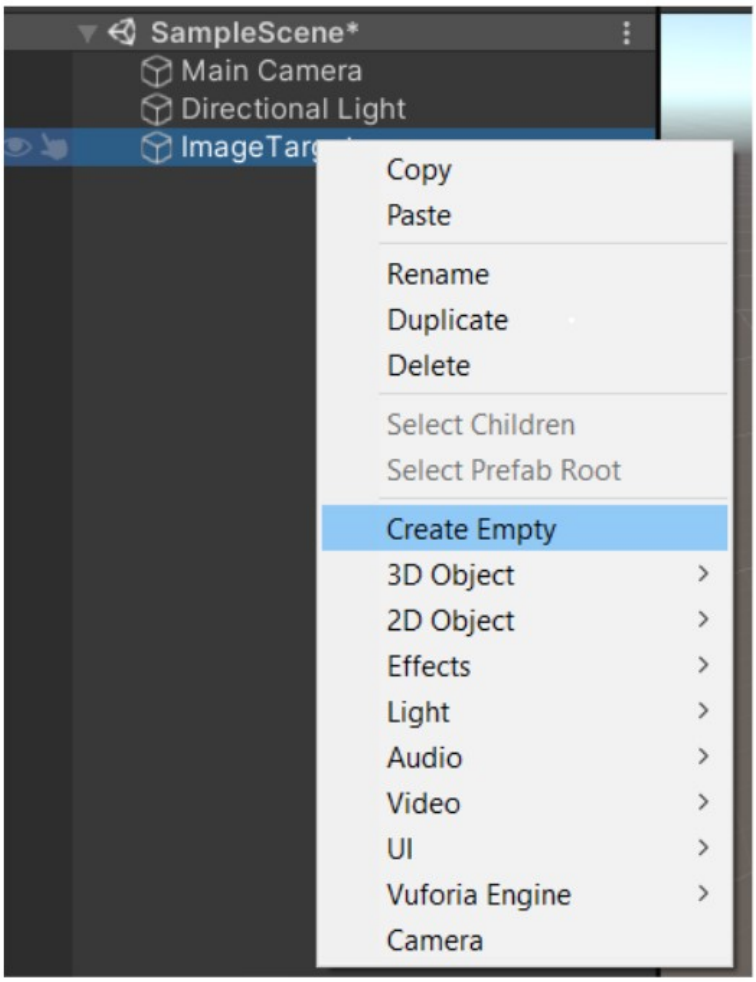
Теперь мы можем переходить к созданию самой дополненной реальности. Для этого в окне иерархии нажмите пкм и в открывшемся контекстном меню выберете Vuforia Engine ->Image Target. Так мы создадим объект маркер.

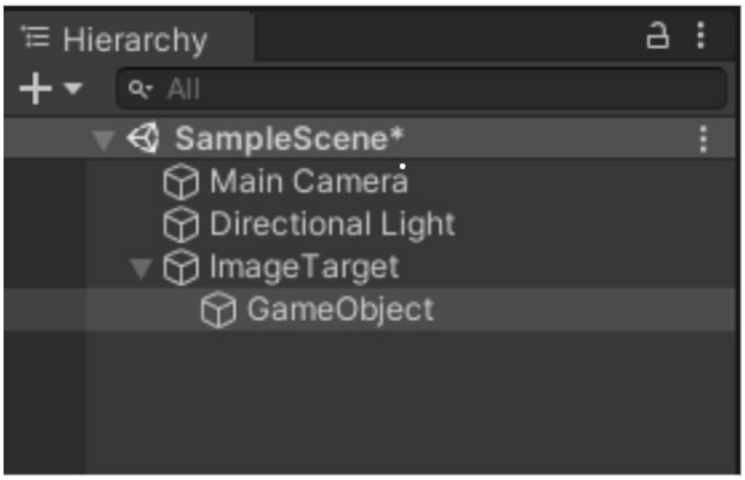


Теперь нам надо выбрать изображение: один раз нажмем на Image Target в окне иерархии — теперь у нас открылся этот же объект и в окне инспектора. В свойстве Image Target Behaviour выберем: тип — из базы данных, саму базу данных и текущее изображение для маркера.

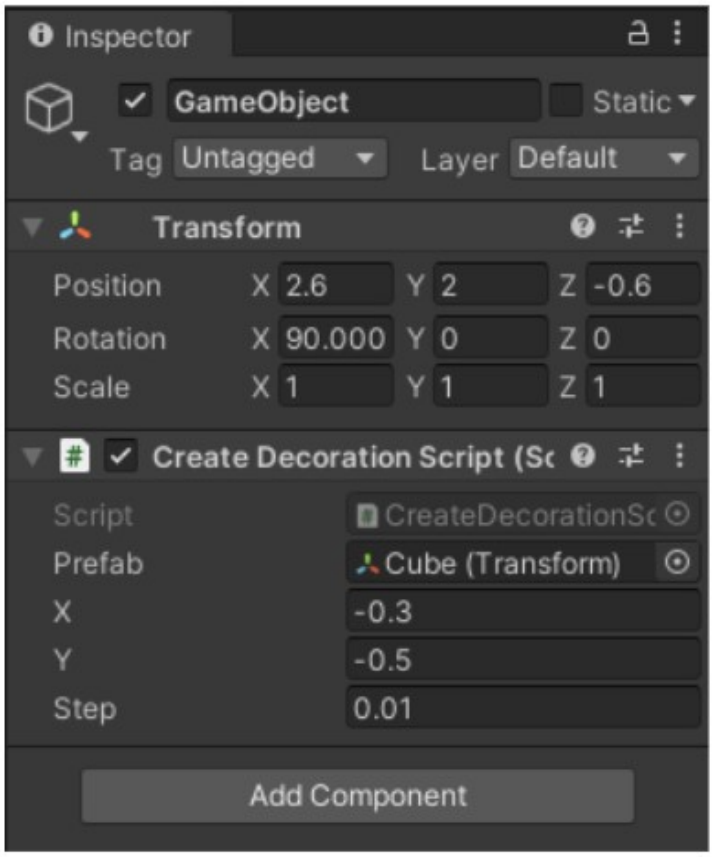


Далее создадим оверлей для маркера. Нажимаем пкм на Image Target и в открывшемся меню выберем создание пустого объекта.





Теперь нам необходимо перетащить на только что созданный объект скрипт CreateOverlayScript, который находится в папке Scripts. И выставить значения у каждого свойства как указано на изображении ниже. Чтобы заполнить свойство Prefab, вам необходимо перетащить объект Cube из папки Prefabs в поле Prefab



На этом все, чтобы протестировать запустите проект и наведите камеру на маркер.

Ответ: 6.

Задача II.1.2.2. Елка (10 баллов)

А вы помните, что скоро новый год? Пора позаботиться о Елке.

Вторая задача направлена на взаимодействие с объектами дополненной реальности через UI.

Вам необходимо собрать AR-приложение, в котором есть 4 отметки, между которыми находятся 3 яруса елки: две усеченных пирамиды — нижний и средний ярусы и одна целая пирамида — верхушка. Также в проекте есть 3D модель звезды, она не является важной частью задачи и нужна просто для красоты и создания новогоднего настроения.

Основная роль приложения — симулятор, в котором можно подбирать размер каждого яруса елки. Размер яруса считается подобранным верно если нижний и верхние грани касаются отметок.

Для изменения размера ярусов используются поля для ввода(InputField). Необходимо чтобы размер ярусов менялся по изменению значения в InputField, а это значит, что нам необходимо взаимодействовать с обработчиком OnValueChange.

В скрипте, который мы предоставляем уже написана функция, меняющая размер ярусов(changeLayerScale) и функция(onValueChange) для обработчика событий полей

ввода. Вам необходимо все правильно подключить.

Проект-заготовка состоит из:

1. скрипт для изменения размера елочного яруса, `/Assets/Scripts/FirBehaviourScript.cs`;
2. префабы + материалы — это части елки, отметки и новогодняя звезда, `/Assets/Prefabs`;
3. картинка, /Assets/A4_task2.jpg.

Заготовка для проекта — https://drive.google.com/file/d/18Vm8umgEIneJ4okWTXn-nqeJKmLmXnRu/view?usp=sharing

Уделите внимание префабам, на сцене — они без материала. Пока вы не прикрепите нужные материалы, вы не сможете точно подобрать ответ — отметка и елочный ярус будут сливаться.

В ответ запишите 3 числа (по возрастанию), значения высоты каждого яруса. Округлите значения до сотых. В качестве разделителя между числами используйте пробел, а между целой частью числа и дробной — запятую.

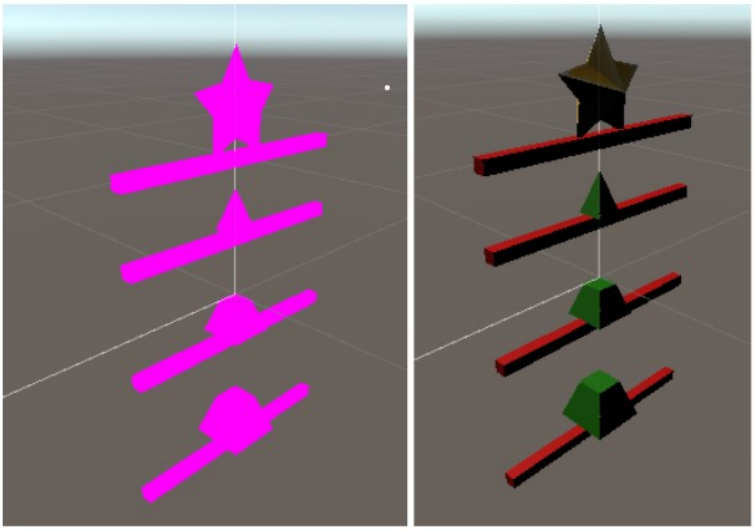
Пример ответа:

0,09 0,23 0,52

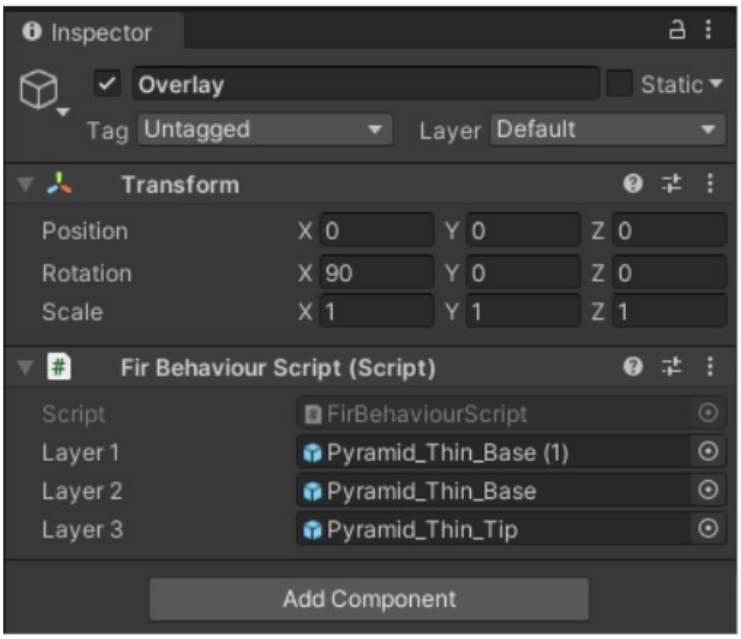
Решение

В данной задаче необходимо все правильно подключить. Если вы посмотрите на заготовку, то увидите что объект — оверлей уже создан, также уже готов канвас на котором располагаются поля для ввода.

Для начала давайте установим материалы для каждого объекта оверлея. В папке Materials находятся 3 материала: зеленый для елочных ярусов, красный для отметок и рождественский для звезды. Чтобы применить материал к объекту достаточно просто перетащить нужный материал на объект в окне иерархии.



Теперь перетащим FirBehaviorScript на объект Overlay. Мы видим что у объекта появились три свойства Layer1, Layer2, Layer3. Заполним эти свойства соответственно уровнями елки.



Последний шаг в настройке оверлея это сделать так, чтобы размер изменялся при вводе значения в поля для ввода. Для этого откроем в инспекторе InputField1, в обработчике события On Value Changed установим объект — Overlay, выберем FirBehaviourScript -> onValueChanged, затем установим InputField1 в поле Input Field.

В результате у вас должно получится так, как представлено на изображении ниже. По аналогии подключите также поля InputField2 и InputField3.

Теперь мы наконец-то можем переходить к дополненной реальности. Подключите вуфорию, вставьте ключ и загрузите изображения для маркеров, как в предыдущем задании. Создайте и настройте Image Target, сделайте Overlay дочерним объектом Image Target. На этом все, можно проверять.

Ответ: 0[.,]03[ ]+0[.,]05[ ]+0[.,]06.

Задача II.1.2.3. Елочка расти (15 баллов)

Третья задача — это усложненная версией второй задачи.

Если в предыдущей задаче вам надо было подобрать значения так, чтобы пирамиды касались красных отметок, то в этой вам необходимо, чтобы пирамиды касались отметок на маркере.

В приложении у вас будут также 3 поля для ввода и 3 яруса елки.

Проект-заготовка состоит из:

- скрипт для изменения размера елочного яруса, /Assets/Scripts/FirBehaviour Script.cs;
- префабы + материалы — это части елки, /Assets/Prefabs;
- картинка, /Assets/A4_task3.jpg.

Заготовка для проекта — `https://drive.google.com/file/d/1om8k9IduNTQ4G58sNmfDeOLmC1CWhq4u/view?usp=sharing`

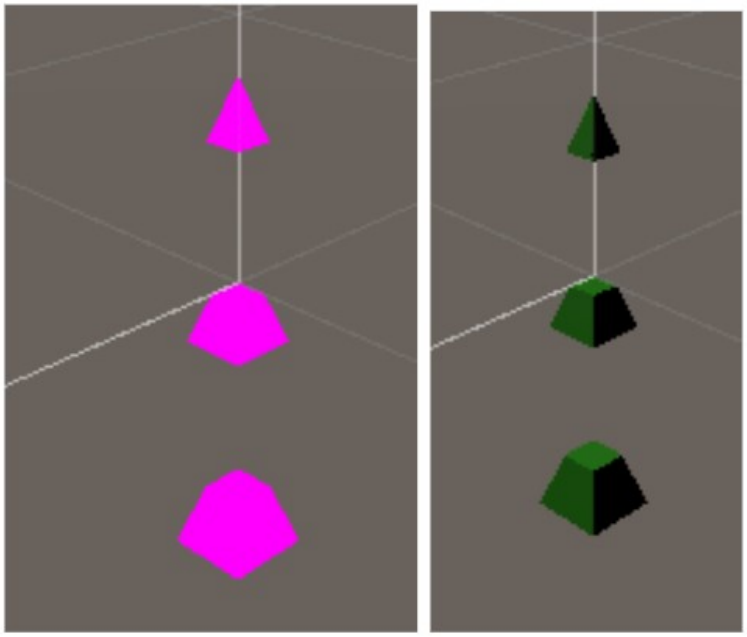
В ответ запишите 3 числа (по возрастанию), значения высоты каждого яруса. Округлите значения до сотых. В качестве разделителя между числами используйте пробел, а между целой частью числа и дробной запятую.

Пример ответа:

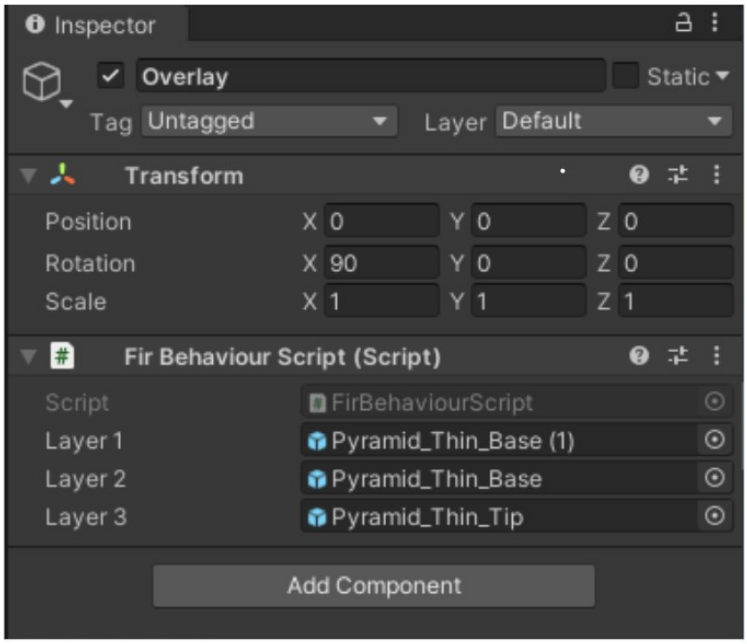
0,09 0,23 0,52

Решение

Как и в предыдущей задаче, сначала установим материалы на объекты.

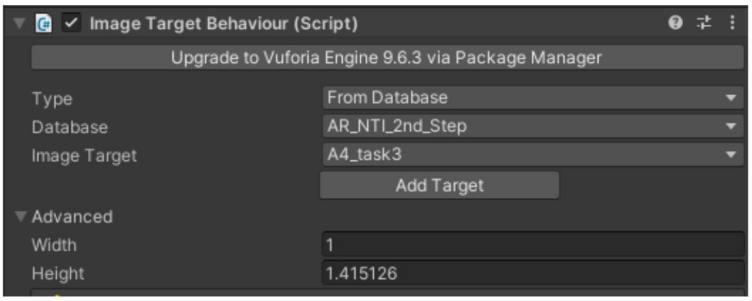


Затем подключаем скрипт и заполним пустые свойства.



Установим скрипт на обработчик событий для полей ввода InputField1, InputField2, InputField3.

Далее подключаем вуфорию, добавляем ключ и маркер, все как в предыдущих заданиях. Но когда будете создавать маркер на сайте обязательно выставите ширину в 1, а когда создадите таргет проверьте что значения ширины и высоты совпадает с изображение ниже. Целью задачи было - показать, что порой неправильные размеры могут привести к провалу проекта.



Ответ: 0[.,]05[ ]+0[.,]07[ ]+0[.,]08.

Задача II.1.2.4. Елки и дома (20 баллов)

Четвертая задача последняя, а значит и самая трудная, не только потому что там надо писать код, но и потому, что надо правильно понять условие. Отведите на

нее больше времени, чем на другие задания из этого блока.

В этой задаче вам необходимо построить вертикальную гистограмму по данным из файла data.json и посчитать количество елок и домов определенного цвета, в так называемом «лесу» (елки изображенные на маркере). Помните, что у гистограммы столбцы равномерно расположены по ширине.

Перед началом решения задачи обязательно посмотрите на данные. Это массив из объектов, где каждый объект состоит из четырех полей: «Id», «IsActive», «Color», «Value».

- «IsActive» показывает надо ли учитывать объект.
- «Color» показывает принадлежность к классу — елки или определенный цвет дома.
- «Value» показывает высоту столбца.

Проект-заготовка состоит из:

- шаблон, /Assets/HistScript.cs
- префабы, /Assets/Prefabs
- картинка, /Assets/A4_task4.jpg
- данные для гистограммы, /Assets/data.json

Заготовка для проекта — https://drive.google.com/file/d/1LgxMFFTnslW4K3-gTelDGfqP1t6J-SJQ/view?usp=sharing

В ответ запишите 3 числа, соответствующих количеству столбцов по ширине полностью пересекающих «лес», касание не учитывать. В качестве разделителя между числами используйте пробел. Расположите числа в следующем порядке:

1. Количество синих домов(столбцов) в лесу.
2. Количество желтых домов(столбцов) в лесу.
3. Количество елок(столбцов) в лесу.

Пример ответа:

1 10 3

Решение

Целью задачи было распарсить JSON, соответственно начнем с разбора кода. Чтобы распарсить json надо создать структуру или класс, в соответствии с которой будет происходить парсинг.

```
64      [System.Serializable]
65      public class ListCols{
66          public col[] col;
67      }
68
69      [System.Serializable]
70      public class col{
71          public string Id;
72          public bool IsActive;
73          public string Color;
74          public float Value;
75      }
76  }
```

Далее мы создаем поля которые будут в классе отвечающем за парсинг.

```
8       public TextAsset jsonFile;
9
10      public GameObject green_prefab;
11      public GameObject blue_prefab;
12      public GameObject yellow_prefab;
13
14      GameObject inst_obj;
15      float y = -1;
16      float z = -1;
```

Мы хотим, чтобы график показывался при нахождении маркера, это значит, что нам достаточно просто выполнять код в функции Start(). Напомним, что данная функция вызывается перед прорисовкой первого фрейма, только если сценарий определен.

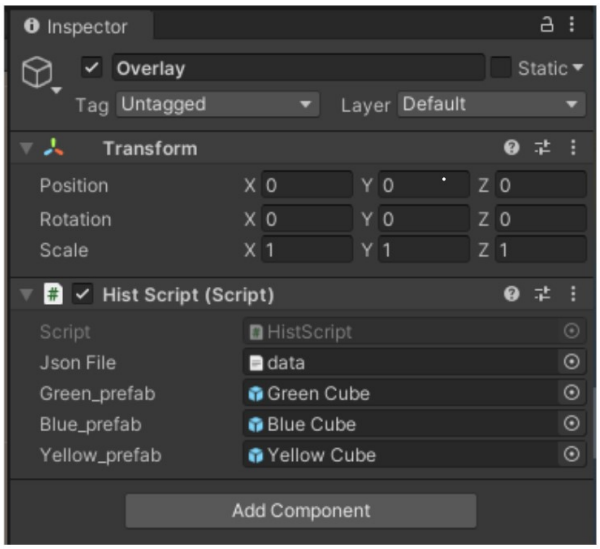
В 22 строчке происходит преобразование json к переменной типа ListCols (это наш кастомный класс), а с 24 по 52 строчку происходит разбор переменной items. Давайте посмотрим на это более детально.

```
18      void Start(){
19          float x = -0.5f;
20          float step = 0.05f;
21
22          ListCols items = JsonUtility.FromJson<ListCols>(jsonFile.text);
23
24  >       foreach (col column in items.col){ ···
60          }
61      }
```

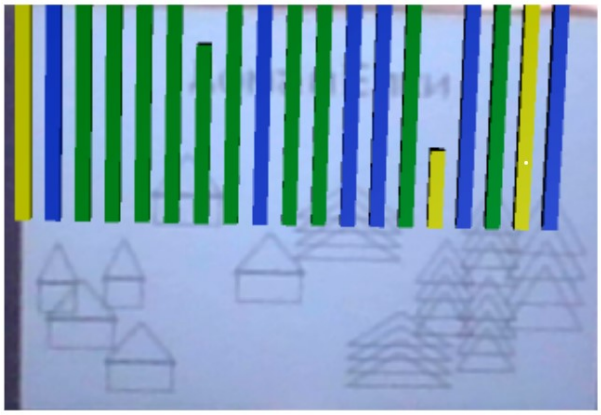
В 24 строке происходит итерация по массиву объектов col. В 25 идет проверка на активность текущего столбца. В 27, 31 и 35 строках происходит разделение по цвету. В 28, 32 и 36 строках создаются новые столбцы на сцене. В 41 мы программно устанавливаем текущий столбец дочерним для объекта оверлея, чтобы график работал только по наведению. Далее с 43 по 49 изменяются координаты и размер текущего столбца.

```
24      foreach (col column in items.col){
25          if (column.IsActive){
26              switch (column.Color){
27                  case "yellow":{
28                      inst_obj = Instantiate(yellow_prefab, new Vector3(x, y, z), Quaternion.identity);
29                      break;
30                  }
31                  case "blue": {
32                      inst_obj = Instantiate(blue_prefab, new Vector3(x, y, z), Quaternion.identity);
33                      break;
34                  }
35                  case "green":{
36                      inst_obj = Instantiate(green_prefab, new Vector3(x, y, z), Quaternion.identity);
37                      break;
38                  }
39              }
40
41              inst_obj.transform.SetParent(gameObject.transform);
42
43              Vector3 addScale = new Vector3(0, 0, column.Value);
44              inst_obj.transform.localScale += addScale;
45
46              MeshRenderer meshRenderer = inst_obj.GetComponent<MeshRenderer>();
47
48              inst_obj.transform.position = new Vector3(x, 0, column.Value / 2f);
49              x += step;
50          }
51      }
```

Наконец-то скрипт готов, надо прикрепить его к объекту и заполнить все поля.



Далее как обычно подключаем вуфорию, ключ, создаем маркер, устанавливаем оверлей дочерним объектом и запускаем.



Так если мы посмотрим справа есть только 4 синих столбика, 2 желтых и 3 зеленых, которые полностью по горизонтали входят в зону елок.

Ответ: 4[,]+2[,]+3.

Компьютерное зрение

Команда AltFuture, взялась за новый проект — разработка системы точной навигации для космического корабля Singularity Bat. Для реализации системы, ученые решили использовать классический подход, связанный с ориентацией корабля по звездам. Алгоритм навигации работает следующим образом:

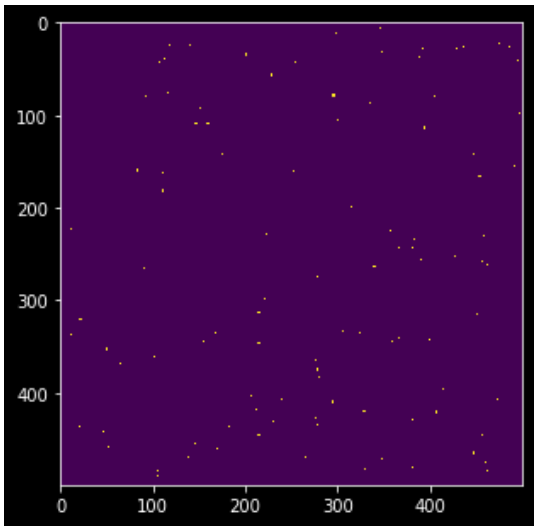
1. Получает координаты звезд в географической системе координат из каталога звездного неба. Эти звезды соответствуют реальным звездам, которые находятся вблизи летательного аппарата.
2. Обработка экрана видения. Внешние камеры корабля отслеживают видимые кораблем звезды. Видимые звезды и звезды из каталога сопоставляются, для того чтобы определить реальные(географические) координаты звезд, видимых кораблем.
3. Далее, по полученным данным, реализуются некоторые математические вычисления, позволяющие сказать точное положение корабля.

Задачи предложенные в рамках олимпиады направлены на решение только одной проблемы из представленных — обработка экрана видения. Первые две задачи — проверяют умение использовать классические алгоритмы компьютерного зрения, в последней задаче предлагается придумать и реализовать свой алгоритм сопоставления звезд.

Задача II.1.3.1. Количество звезд (10 баллов)

На вход поступает одно изображение. Ваша задача определить сколько на изображении звезд.

Пример входных данных:



Пример выходных данных:

78

Ссылка на пример данных:

https://drive.google.com/file/d/1n10rQ7YGa81mRRgc4ZSg5_wBszONDJMr/view?usp=sharing

Для решения задачи, вам будет выдана ссылка на гугл-диске с файлом, для которого нужно найти решение.

Решение

Для решения этой задачи можно воспользоваться алгоритмом связных компонент. Его реализация уже есть во многих языках, а потому писать самостоятельно необязательно.

Алгоритм состоит из трех этапов:

1. Раскраска элементов. На вход подается бинаризированная картинка, на выходе та же картинка, но каждый объект на которой окрашен в свой уникальный цвет.
2. На данном этапе алгоритм ранжирует цвета от 1 до N, где N кол-во объектов.
3. Реализуется интерфейс, с помощью которого по индексу-цвету объекта, можно получить его параметры на картинке: площадь, периметр и т. д.

В питоне пункты 1 и 2 реализует функция label, а пункт 3 функция regionprops. Рассмотрим как будет выглядеть реализация алгоритмы в коде.

```python
# импорт библиотеки
from skimage.measure import label, regionprops
```

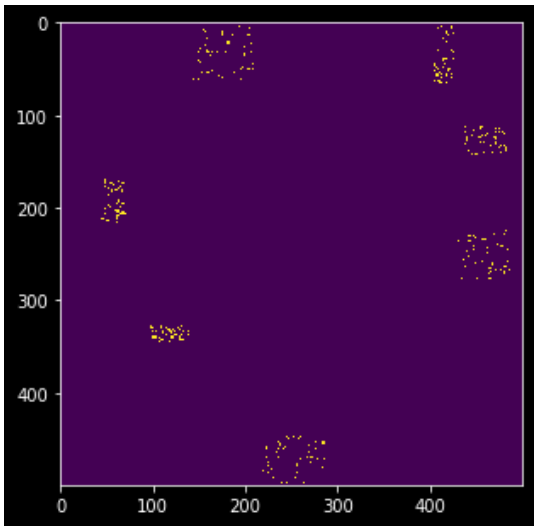
```python
# загрузка изображения и его визуализация
img = plt.imread("tasks/1/5.png")

plt.figure(figsize = (5,5))
plt.imshow(img)
```

```python
# реализация алгоритма
LB_image1 = label(img)
points_imgage1 = regionprops(LB_image1)
print(len(points_imgage1))
```

Задача II.1.3.2. Созвездия (15 баллов)

Одной из задач, которую часто требуется решить для локализации космических кораблей — это нахождение на небе определенных созвездий. В рамках олимпиады эта задача была упрощена. По данному космическому снимку требуется определить количество созвездий и их площадь.

За созвездие считаются точки, между которыми расстояние меньше некоторого порогового. Чаще всего это расстояние ученые подбирают сами, исходя из того, сколько кластеров они видят на глаз. Пример входных данных:



Пример выходных данных: 7, 680, 1080, 1276, 1302, 2800, 3283, 3705

Ссылка на пример данных:

https://drive.google.com/file/d/1l5Nwyh_4xOTLEuu9J2LYbqN6adhdaCqN/view?usp=sharing

Здесь 7 — число найденных кластеров. Дальнейшие числа площади кластеров отсортированные в порядке возрастания. Площадь считается по формуле:

$$width = max(X) - min(X)$$

$$height = max(Y) - min(Y)$$

$$S = width \cdot height;$$

Где X и Y — наборы координат точек-звезд каждого из кластеров.

Для решения данной задачи, полезно воспользоваться методами sklearn-clustering. Или методами встроенными в opencv. Возможна и самостоятельная реализация алгоритма с помощью некоторых математических методов.

Для решения задачи, вам будет выдана ссылка на гугл-диске с файлом, для которого нужно найти решение.

Решение

Для решения данной задачи предлагается воспользоваться алгоритмом кластеризации. Так как нам неизвестно сколько кластеров будет на изображении, то использовать имеет смысл алгоритм, который использует в качестве метрики плотность объектов друг к другу. В данном решении использовался алгоритм Agglomerative Clustering. Подробнее про метод можно почитать здесь — https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.

```
# загрузка изображения
index = 2
img = plt.imread("tasks/2/%d.png"%index)
img = plt.imread("9.png")
plt.figure(figsize = (5,5))
plt.imshow(img)
```

Для начала нам нужно сгенерировать обучающую выборку. Для этого нужно найти центры положения всех объектов на снимке. В этом помогут также методы label и regionprops.

```
LB_image1 = label(img) # раскрашиваем объекты на изображении
points_imgage1 = regionprops(LB_image1) # получаем параметры объектов
X = [r.centroid for r in points_imgage1] #получаем центры положения объектов
print(len(X))
```

Далее обучаем модель. Для того, чтобы модель работала правильно, параметр distance_threshold = 160. Это число было подобрано эмпирическим путем.

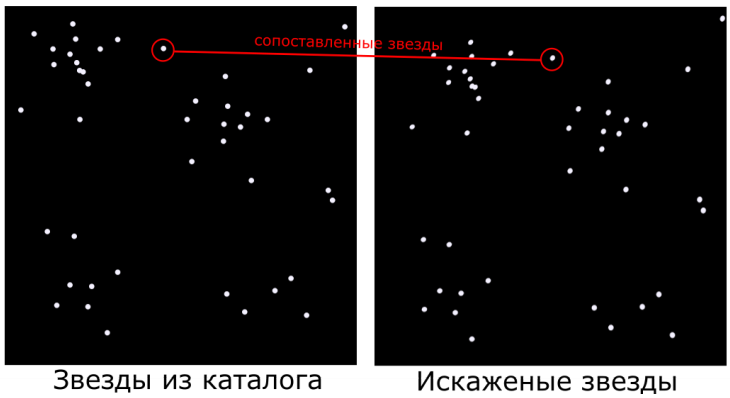
```
from sklearn.cluster import AgglomerativeClustering
import numpy as np

clustering = AgglomerativeClustering(n_clusters=None, distance_threshold  = 160).fit(X)

labels = clustering.labels_
print(labels)
```

Теперь, когда мы кластризовали объекты, мы можем посчитать площадь, которую занимает каждый класс, по формуле указаной в задании.

```
X = np.asarray(X)
areas = []
for i in (list(set(labels))):
    mask = np.where(labels == i)
    plt.scatter(X[:,0][mask], X[:,1][mask])
    mx_w = int(np.max(X[:,1][mask])-np.min(X[:,1][mask]))
    mx_h = int(np.max(X[:,0][mask])-np.min(X[:,0][mask]))
    areas.append(mx_w*mx_h)

answer = [len(set(labels))] + sorted(areas)
print(answer)
```

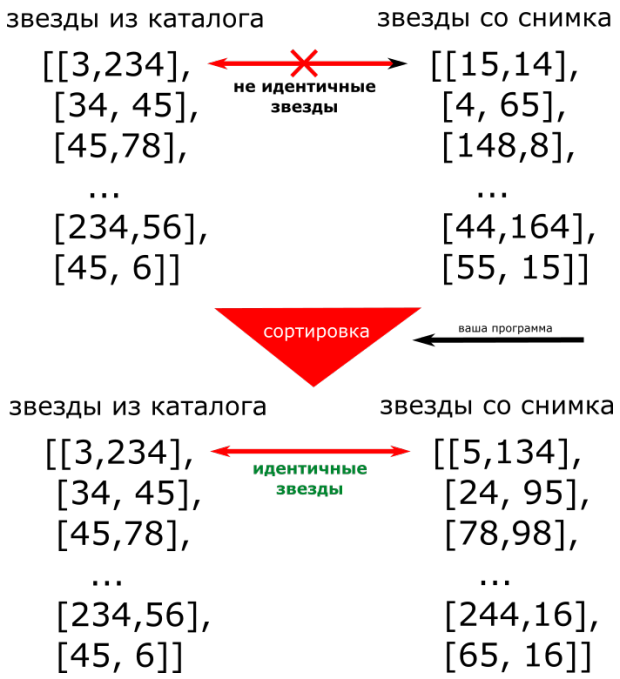
Задача II.1.3.3. Сопоставление звезд (25 баллов)

Проблема при сопоставлении звезд чаще всего заключается в том, что из-за приборов оптики снимок искажается и поэтому видимые звезды не будут иметь ту же координату что, звезды в каталоге.



Ваша задача — сопоставить точки, найденные на двух изображениях. Чтобы избежать неточностей, на вход будут подаваться не изображения, а именно координаты звезд в виде двух массивов. Первый массив — звезды из каталога, второй массив — звезды видимые кораблем(звезды со снимка).

На выход нужно передать второй массив, отсортированный в соответствии с первым. Т. е. если мы возьмем первую точку массива из звезд каталога, и первую точку из массива видимых звезд — это считается одна и та же звезда.



Для решения данной задачи можно рассмотреть различные подходы. Некоторые алгоритмы сопоставления звезд описаны в интернете, но возможна реализация и собственного алгоритма.

Пример данных для задачи можно найти по ссылке: https://drive.google.com/file/d/1Ckj3tVDWvV7xO__eAzZBTbOmrglnIuWr/view?usp=sharing и ответ к ним: https://drive.google.com/file/d/17uLfOOmcrbISoZT64J7fusZBDGbNj1yM/view?usp=sharing

Для решения задачи, вам будет выдана ссылка на гугл-диске с файлом, для которого нужно найти решение.

Пример входных данных:

[[455, 182], [230, 412], [32, 437], [237, 185], [147, 3],...,[205, 306], [366, 145], [214, 487], [104, 312], [158, 373], [37, 233]]

[[407, 353], [208, 136], [102, 263], [186, 295], [127, 510],...[46, 151], [368, 86], [60, 302], [502, 394], [298, 392], [298, 392]]

Выходные данные:

[[469, 199], [258, 475], [71, 518], [263, 219], ..., [385, 164], [244, 561], [138, 372], [190, 437], [74, 288]]

Решение

К решению данной задачи можно было подойти с разных сторон. Авторский метод предлагает свой собственный алгоритм данной задачи.

Для алгоритма используется специальная структура данных Star. Структура содержит 5 полей.

index — индекс звезды; position — положение звезды в декартовой системе координат; соседи — ссылка на другие звезды, представленные в структуре данных Star; rots — массив углов между положением звезды цели и положением звезды соседей.

Алгоритм:

1. Сначала инициализируются массивы stars_1 и stars_2 — они хранят данные о звездах, отформатированные в структуру данных Star.
2. Инициализируется массив того же размера, что и stars_1, в нем будут храниться сопоставления между звездами: в каждой ячейке индекс звезды из списка stars_2, который лучше всего подходит для звезды из списка stars_1.
3. Затем, просматривая звезды между собой, ищутся звезды с максимальной метрикой LCS_DYN. В свою очередь, алгоритм LCS_DYN — это алгоритм поиска самой длинной общей подпоследовательности.
4. Он сравнивает списки углов к звездам из двух списков.

Другими словами, гипотеза алгоритма: соответствующие звезды с двух изображений будут иметь самую длинную общую подпоследовательность в отсортированном списке углов между целевой звездой и соседними.

Необходимы методы для работы алгоритма:

```python
#координаты звезд на снимке
def get_stars_coords(points):
    coords = []
    for point in points:
        coords.append(point.centroid)
    return coords

def fill_dyn_matrix(x, y):
    L = [[0]*(len(y)+1) for _ in range(len(x)+1)]
    for x_i,x_elem in enumerate(x):
        for y_i,y_elem in enumerate(y):
            if abs(x_elem - y_elem)<20:
                L[x_i][y_i] = L[x_i-1][y_i-1] + 1
            else:
                L[x_i][y_i] = max((L[x_i][y_i-1],L[x_i-1][y_i]))
    return L
```

```python
#алгоритм нахождения максимальной подпоследовательности
def LCS_DYN(x, y):
    L = fill_dyn_matrix(x, y)
    LCS_x = []
    LCS_y = []
    indx_x = []
    indx_y = []
    x_i,y_i = len(x)-1,len(y)-1
    while x_i >= 0 and y_i >= 0:
        if abs(x[x_i] - y[y_i])<10:
            LCS_x.append(x[x_i])
            LCS_y.append(y[y_i])

            indx_x.append(x_i)
            indx_y.append(y_i)
            x_i, y_i = x_i-1, y_i-1
        elif L[x_i-1][y_i] > L[x_i][y_i-1]:
            x_i -= 1
        else:
            y_i -= 1
    LCS_x.reverse()
    LCS_y.reverse()
    return LCS_x, LCS_y
```

Структура данных:

```
class Star:
    def __init__(self, indx, x, y):
        self.indx = indx
        self.position = (x,y)
        self.neighbours = []
        self.rots = []
        self.distance = []
        self.nearest = []
        self.nearest_indx = []
       # self.__init_neighbours()

    def __str__(self):
        return "idx = %d, x = %d, y = %d"%(self.indx, self.position[0], self.position[1])

    def init_neighbours(self, stars):
        self.neighbours = np.asarray(sorted(stars, key = lambda s: distance.euclidean(self.position,s.position)))
        self.rots = [math.degrees(math.atan2(self.position[0] - s.position[0],
                                             self.position[1] - s.position[1])) for s in self.neighbours ]
        self.distance = [distance.euclidean(self.position, s.position) for s in self.neighbours]
        self.distance = np.asarray(self.distance)

    def nearest_neighbours(self):
        std = np.std(self.distance)
        thresh = np.mean(self.distance) - 1.2*np.std(self.distance)

        nearest = np.where(self.distance<thresh)
        self.nearest = self.neighbours[nearest]
        self.nearest_indx = [n.indx for n in self.nearest]
        return self.neighbours[nearest]
```

```
def activate_stars(stars):
    for s in stars:
        s.init_neighbours(stars)
        s.nearest_neighbours()
```

Получение звезд со снимка и перевод их в структуру Stars

```
stars_coords_1 = np.asarray(eval(task[0]))
stars_coords_2 = np.asarray(eval(task[1]))
```

```
from scipy.spatial import distance
from scipy.optimize import linear_sum_assignment
```

```
stars_1= [Star(i, stars_coords_1[i][0], stars_coords_1[i][1]) for i in range(len(stars_coords_1))]
stars_2= [Star(i, stars_coords_2[i][0], stars_coords_2[i][1]) for i in range(len(stars_coords_2))]

activate_stars(stars_1)
activate_stars(stars_2)
```

Реализация самого алгоритма

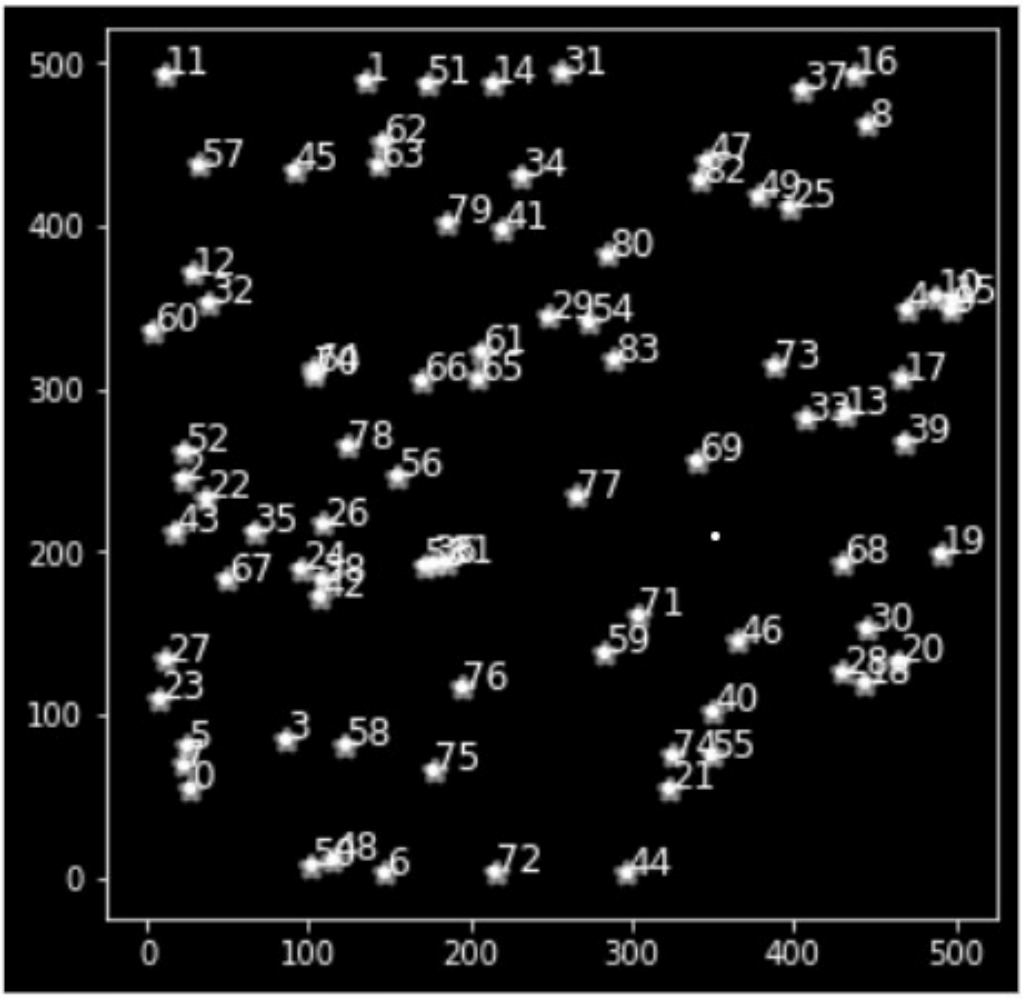
```
idnx = [-1]
distanc = [1000]

for i in range(0,len(stars_1)):
    max_len = 0
    for j in range(0,len(stars_2)):
        X, Y = LCS_DYN(stars_1[i].rots, stars_2[j].rots)
        dist_xy  = distance.euclidean(X,Y)
        if len(Y)>=max_len :
            idnx[i] = j
            distanc[i] = len(Y)
            max_len = len(Y)
    idnx.append(-1)
    distanc.append(1000)
```

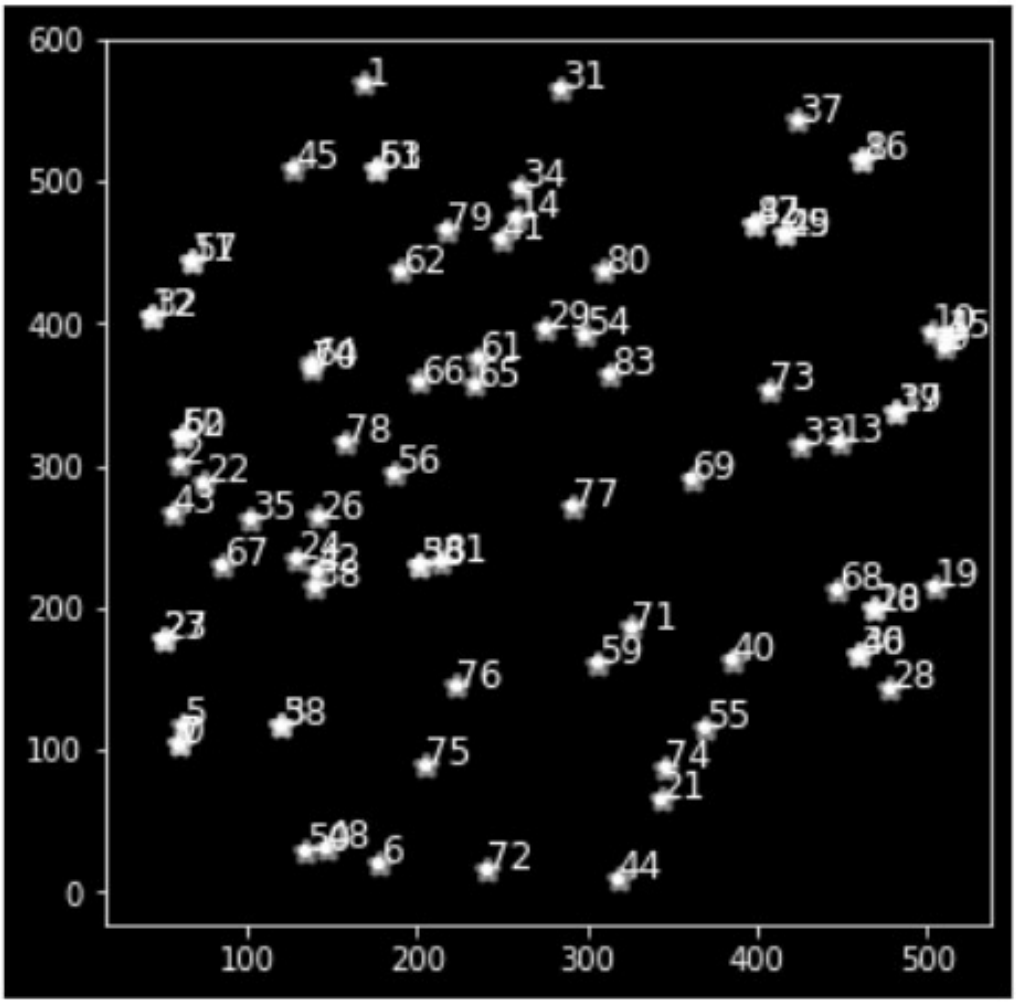
Убираем неправильно сопоставленные звезды

```
indx_1 = [s.indx for s in stars_1]
std = np.std(distanc[:-1])
thresh =  np.mean(distanc[:-1]) - std

mean_thresh = int(len(indx_1)*0.6)
indx_ = sorted(list(zip(indx_1,idnx,distanc)), key = lambda e: e[-1])[::-1]

last_indx = np.where(np.asarray(indx_)[:,2] > thresh)[0][-1]
indx_1 = np.asarray([i[0] for i in indx_])
indx_2 = np.asarray([i[1] for i in indx_])

for i in range(1,len(indx_1)):
    if len(np.where(indx_1==i)[0])>1:
        indx_1[indx_1==i] = -1
        indx_2[indx_2==i] = -1
```

Визуализация результата

```
from plotter import plot_sky_image

print("Звезды из каталога")
#stars_catalog = np.append(stars_catalog[:8], stars_catalog[:8])

plot_sky_image(stars_coords_1[indx_1][:last_indx], text = True)
print("Звезды со снимка")

#print(len(stars_image))
plot_sky_image(stars_coords_2[indx_2][:last_indx], text = True)

#print(accuracy(stars_coords_1[indx_1][:last_indx], stars_coords_2[indx_2][:last_indx]))
```


500
400
300
200
100
0
0
100
200
300
400
500
11
1
51
14
31
37
16
8
62
57
45
63
34
47
82
49
25
79
41
80
12
32
60
29
54
61
83
73
17
66
65
33
13
39
52
78
56
69
22
77
43
35
26
68
19
67
24
71
59
46
30
27
76
23
40
3
58
5
75
74
55
21
6
72
44



Командная часть

Данный модуль состоит из 2 задач на выбор, т. е. вы можете выбрать и решать как одну из них, так и обе. В качестве ответов, пожалуйста, прикладывайте только ссылки на ресурсы, запрашиваемые в задании. Для видео-демонстраций мы написали свой сценарий, но вы можете и отходить от него. Обратите внимание, что видео не должно быть длинее 3 минут.

Задача II.2.1. Картинная галерея (30 баллов)

В данной задаче вам необходимо реализовать сервис, который позволяет отслеживать количество просмотров картинной галереи в реальном времени и строить инфографику по полученным данным. Т.е. при наведении на маркер считается количество пользователей, которые прямо сейчас также наводят на этот маркер, и строятся графики, по информации, которую пользователи указывали при регистрации.

Ссылка на примеры экранов приложения — https://drive.google.com/drive/folders/1pzy4e2-lGB5AWiC2sNkQYakxX2ODrKfx?usp=sharing

При регистрации необходимо указать минимум 3 параметра. Вы можете восполь-

зоваться теми, что мы предлагаем или же придумать свои, но в таком случае стоит учитывать формат данных, они должны подходить для построения инфографики. Наши параметры можно увидеть в примерах экранов приложения.

Данный сервис состоит из серверной части и приложения. Сервер принимает следующие запросы:

- Регистрация пользователя. (POST запрос отправляется один раз при входе) Вам необходимо сделать регистрацию при входе в приложение, как в анонимных чатах, т. е. при закрытии приложения пользователь заново регистрируется, но при уходе (перемещении) в фон данные сохраняются, и новая регистрация не требуется. Соответственно, при закрытии приложения должно быть реализовано удаление пользователя.
- Регистрация и получение количества текущих просмотров. (POST запрос отправляется при наведении на маркер, а затем с определенным периодом при сохранении наведения на маркер) При наведении отправляется запрос, который обновляет количество просмотров текущего маркера, и, соответственно, если маркер перестает отслеживаться, то и количество просмотров изменяется. Предполагается, что POST запрос возвращает в ответе текущее количество просмотров.
- Получение данных о текущем пользователе. (GET запрос отправляется при наведении и нажатии на кнопку, а затем с определенным периодом при сохранении наведения на маркер) После получения текущих просмотров должна быть возможность посмотреть на статистику по людям, которые в данный момент наводят на маркер. Например, если мы наводим на маркер и нажимаем на кнопку «Показать графики», на экране строятся 3D графики в дополненной реальности в зависимости от того, какие данные вы запрашиваете при регистрации.

Определенный период — это задержка между запросами. Она нужна, чтобы не спамить постоянно данными на сервер. Вы сами выбираете сколько она будет длиться.

Хранение данных реализуйте на ваше усмотрение. Вы можете хранить данные как в файле, так и в базе данных. Мы рекомендуем остановиться на хранении json файлов для более простого взаимодействия с сервером.

Приложение отправляет запросы, получает ответы от сервера и строит инфографику по данным, полученным от сервера.

Поскольку основная цель сервиса — отслеживание просмотров картин, вам необходимо использовать несколько различных маркеров, каждый из которых соответствует только одному произведению. В нашем примере это следующие картины: «Девятый вал» Айвазовского, «Звездная ночь» Ван Гога и «Мона Лиза» Леонардо Да Винчи. Вы сами решаете сколько и каких изображений выбрать для создания маркеров, но их должно быть не менее 3.

Обязательное условие — инфографика должна быть в 3D в дополненной реальности. Если она будет в 2D или не в дополненной реальности, баллы будут снижены.

Сервер можно написать на Python с использованием веб-фреймворка Flask, ссылка на официальную документацию

Задача будет оцениваться по следующим критериям:

		Критерий	Максимальное количество баллов
1		Реализован сервер, все запросы работают корректно	10
	1.1	Регистрация пользователя	3
	1.2	Регистрация и получение количества текущих просмотров	3
	1.3	Получение данных о текущем пользователе	3
	1.4	Вся система работает вместе	1
2		Реализовано приложение, приложение принимает данные	5
3		Работает функция отслеживания просмотров	5
4		Присутствует экран с формой регистрации	2
5		Есть инфографика по данным с сервера	3
6		Инфографика выполнена в AR в 3D	5
Итого			30

В качестве ответа приложите ссылки на следующие ресурсы:

- код на github/gitlab/bitbucket;
- видео демонстрацию работы приложения (максимум 3 минуты);
- установочный файл.

Пример сценария для видео-демонстрации

При записи видео должны быть учтены все критерии, если что-то не будет показано или показано не в полном объеме, то баллы будут снижены.

1. Для начала покажите регистрацию одного пользователя.
2. Наведите устройство на маркер. Ожидаемое поведение — один просмотр.
3. Продемонстрируйте построение графиков для одного устройства.
4. Зарегистрируйте второго пользователя на другом устройстве. Не убирайте первое устройство.
5. Продемонстрируйте наведение двух устройств на один маркер. Можно сделать копию маркера и наводить сразу и на маркер и на его копию разными устройствами, если вам так удобнее. Ожидаемое поведение — два просмотра.
6. Продемонстрируйте построение графиков для двух устройств на одном маркере или его копиях.
7. Продемонстрируйте наведение двух устройств на два различных маркера. И построение графиков. Ожидаемое поведение — при наведении на 2 разных маркера должно быть по одному просмотру на каждом из них и различные графики.

Задача II.2.2. Трансляция игры в AR (30 баллов)

Многие из вас видели окно «Нет подключения к Интернету» при проблемах с сетью в Google Chrome. Но знали ли вы, что можно превратить это сообщение в игру с динозавром? Ваша задача — создать сервис, позволяющий отслеживать передвиже-

ние динозавра и препятствий на его пути, и при наведении на маркер транслировать их в приложении с дополненной реальностью в 3D.

Если у вас установлен Google Chrome то, вам достаточно набрать в адресной строке chrome://dino/, в противном случае вы можете открыть игру на стороннем сайте, например здесь T-rex game (https://googledino.ru/).

Ссылка на примерные экраны приложения — https://drive.google.com/drive/folders/1qflpBzVz5pyGr1ZbbiPX4WwRhdE0WvvH?usp=sharing

Данный сервис состоит из трекера игры, сервера и приложения.

Трекер игры следит за передвижением игровых объектов и сохраняет их координаты. Мы рекомендуем делать эту часть на Python, но вы можете выбрать любой другой язык программирования. Посмотрите на разные форматы передачи данных, выберите для себя наиболее удобный и подходящий в рамках задачи. Вы сами выбираете что и в каком формате сохранять, главное, чтобы вы потом могли это обработать.

Сервер получает данные об игровых объектах и отдает их в приложение. Сервер работает со следующими запросами:

- Отправка данных с состоянием игровых объектов. (POST запрос отправляется с определенным периодом) Данные от трекера отправляются на сервер в удобном для вас формате.
- Получение данных об игровых объектах (GET запрос отправляется с определенным периодом) Получение данных с сервера об изменении игровых объектов для работы приложения.

Определенный период — это задержка. Она нужна, чтобы трансляция корректно работала. Например, трекер записывает данные о передвижении игровых объектов в течении 10 секунд. Затем отправляет их на сервер. Данные с сервера принимает мобильное приложение, которое отправляет запросы например раз в 5 секунд и отображает их.

Приложение воссоздает игру по координатам игровых объектов, при наведении на маркер. Иными словами при наведении происходит соединение с сервером и начинается трансляция игры с задержкой.

Также предусмотрите возможность показать, что игра закончилась и динозавр врезался в кактус.

Сервер можно написать на веб-фреймворке Flask, ссылка на официальную документацию — https://flask.palletsprojects.com/en/1.1.x/

Для реализации трекера посмотрите библиотеки :

- skimage или opencv — обработка изображений;
- mms — быстрое взятие скриншотов рабочего стола;
- opencv или matplotlib — для вывода отладочной информации.

Не рекомендуем использовать opencv на C#.

Задача будет оцениваться по следующим критериям:

		Критерий	Максимальное количество баллов
1		Реализован трекер игровых объектов	10
2		Реализован сервер, все запросы работают корректно	6
	2.1	Отправка данных с состоянием игровых объектов	3
	2.2	Получение данных об игровых объектах	3
	2.3	Вся система работает вместе	1
4		Реализовано приложение, работает получение данных с сервера	3
5		Готовы 3D модели	3
6		Динозавр прыгает через кактусы, где кактусы рандомно генерируются	2
5		Есть трансляция	5
Итого			30

В качестве ответа приложите ссылки на следующие ресурсы:

- код на github/gitlab/bitbucket;
- видео демонстрации работы приложения (максимум 3 минуты);
- установочный файл.

Пример сценария для видео-демострации

При записи видео должны быть учтены все критерии, если что-то не будет показано или показано не в полном объеме, то баллы будут снижены.

1. Продемонстрировать подключение к серверу в приложении.
2. Показать вместе, как человек одновременно играет и как происходит трансляция при наведении на маркер. Должно быть четко видно прыжки динозавра, и то что он не соприкасается с другими объектами.
3. Соприкосновения видны только при выводе сообщения Game Over. Показать сообщение о конце игры.