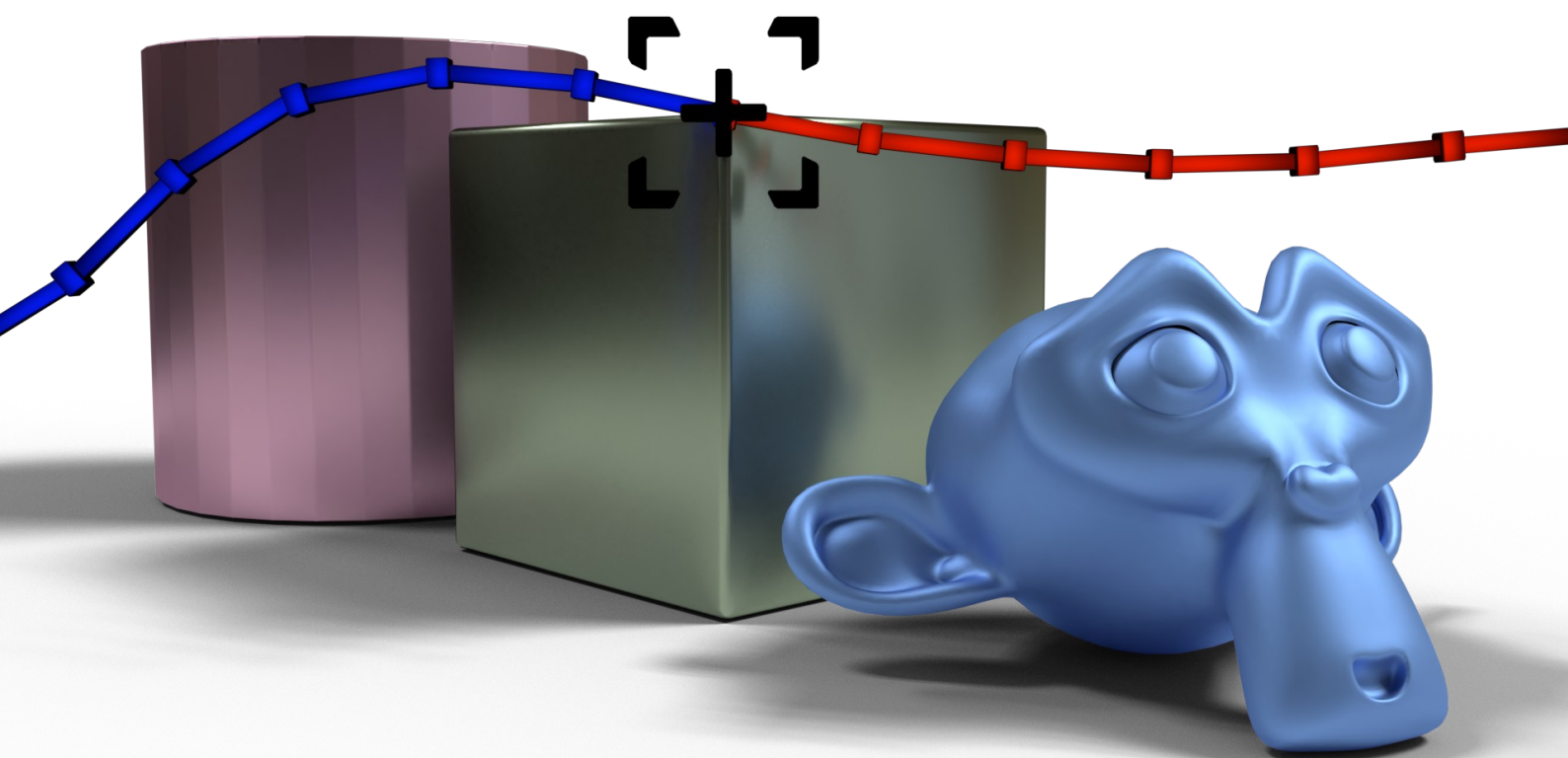


BLENDER: СОВМЕЩЕНИЕ РЕАЛЬНОГО С ВИРТУАЛЬНЫМ

Денис Скиба aka Mapper720
<http://mapper720.ru>



29.10.2017

Что тут есть

От автора.....	3
Съёмка видео.....	4
Окно Motion Tracking в Blender'e.....	5
Как устроен маркер.....	7
Свойства маркера.....	8
Как добавлять маркеры.....	9
Вручную или автоматически?.....	11
Трекинг.....	12
Фотограмметрия.....	14
Для чего нужна опция Tripod?.....	16
Для чего нужна опция Keyframe?.....	16
Создание виртуальной камеры.....	16
Информация о камере.....	17
Что делать, если solve error слишком высокий?.....	19
SetFloor, SetOrigin, SetX/Y.....	22
Проверка.....	23
Советы, лайфхаки.....	28
Освещайте сцену с помощью HDRI-карты.....	28
Не поленитесь допилить модели.....	30
Чёткость и размытие.....	30
Использование спикеров для озвучки.....	31

В самых общих чертах — Motion Tracking есть технология, позволяющая объединить компьютерную модель с реальной съёмкой. Чтобы было лучше понятно, о чём идёт речь — посмотрите, например, видеоролик [«City 22701000 Revisited»](#). Или [«Welcome to City 89701000»](#).

Уроков по данной теме на том же YouTube не так уж мало. Есть и русскоязычные, и иностранные. Но изучение трекинга по данным урокам связано с рядом сложностей, а именно:

0. Слишком долго. Общая длительность этих уроков составит никак не менее 10 часов, а, вероятно, и не один десяток.
1. Не даётся теоретической основы. Проще говоря, пользователю показывают, как что-то сделать и что из этого получится, но не объясняют, **как** оно работает, и **почему** получается именно то, что демонстрируется на экране. Что в результате? Если у человека, посмотревшего видеоурок, что-то пойдёт не так, как запланировано, он вряд ли сможет найти и устранить причины.
2. Английским языком тоже, очевидно, владеют не все, особенно там, где воспринимать нужно устную речь.

Несмотря на п. 0, посмотреть видеоуроки на том же YouTube стоит. На что обратить внимание?

0. На канале некоего Александра Журавлёва имеется несколько уроков по теме — вполне неплохо излагает основы. [Раз, два, три, четыре](#).
1. Для владеющих английским - урок [«How to do Camera Tracking in Blender»](#) от Oliver Villar (около двух часов). Автор не только рассказывает и показывает, как что-то сделать, но и объясняет некоторые принципы работы, позволяющие лучше понять, что к чему (в русскоязычных уроках, как мне показалось, с такими объяснениями похуже).
2. Курс [«Track, Match, Blend!»](#). Длинный. На английском. Для продвинутых.
3. [«Track, Match, Blend! 2»](#). Продолжение предыдущего. Чтобы посмотреть, придётся оплатить подписку на Blender Cloud (мы ведь не против поддержать Blender Foundation?). Мне показалось, он попроще предыдущего и явно короче. Объясняется вроде как более доступно. Хотя это субъективно.

Здесь я постараюсь объяснить базовые принципы работы с трекингом в программе Blender, обобщив те знания, что были получены из различных уроков и обсуждений в тематических сообществах. Конечно, всего я знать и описать здесь не могу, поэтому свои замечания и дополнения (ежели таковые возникнут) — пишите на

admin@mapper720.ru

Примечание. Актуальная на момент написания данного пособия версия Blender — 2.79.

Прежде, чем что-либо делать с видео в Blender'e, нужно, как ни странно, это видео заполучить. Лучший вариант — снять самому. Этим вы избавите себя от возможных проблем с авторскими правами (при использовании чужого видео), а также обеспечите себе возможность снимать именно так, как требуется вам для реализации вашей задумки.

Итак, исходное видео снимаем самостоятельно. Здесь стоит принять во внимание несколько правил:

0. На видео должны присутствовать объекты, выделяющиеся на окружающем фоне. Желательно, чтобы оные были распределены по всему кадру, а не находились лишь в одной области.



Именно за эти точки будет «цепляться» Blender. **Важно:** эти объекты должны быть неподвижными! Пример плохой сцены для видеосъёмки — покрытое белым снегом поле.

1. Желательно, чтобы камера не двигалась слишком быстро — особенно если снимает с не очень большой частотой кадров в секунду. Это приведёт к смазыванию точек, указанных в п. 0, что может существенно осложнить дальнейшую работу.
2. Стоит заранее определиться, что и куда мы будем встраивать.
3. Между камерой и той областью, куда планируется добавить новый объект, не должно быть никаких препятствий. Оные осложнят (возможно, серьёзно) дальнейшую работу. Здесь между камерой и встроенным объектом нет никаких препятствий:



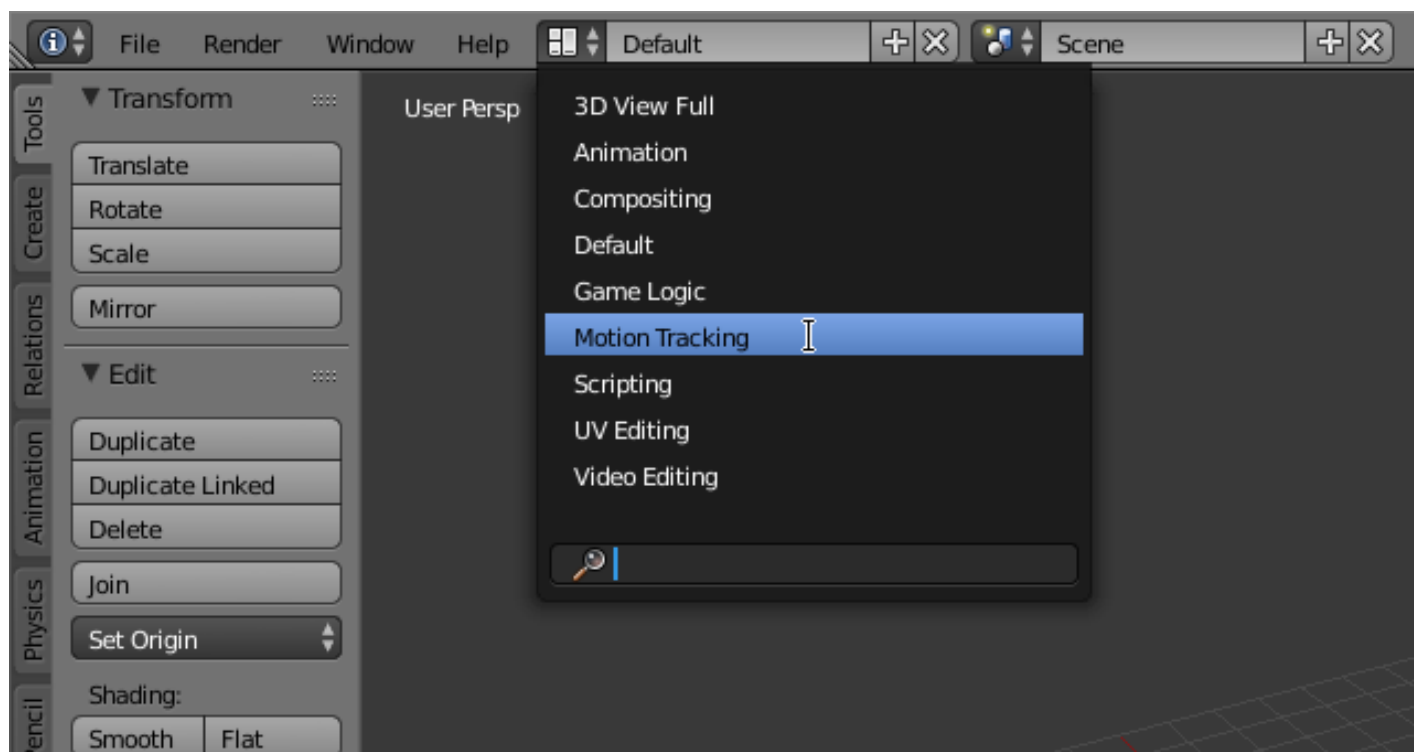
А здесь ворота, которые, очевидно, создадут проблемы:



Теоретически, проблема решаема. Можно замоделировать виртуальную копию ворот, которая будет перекрывать ворота на видео и удалять эту область из кадра. Однако создание такой виртуальной копии и соотнесение её с воротами на видео — дело долгое, сложное и тоскливое, так что лучше избегать этого, если, конечно, по задумке автора объект не должен быть именно **позади** ворот.

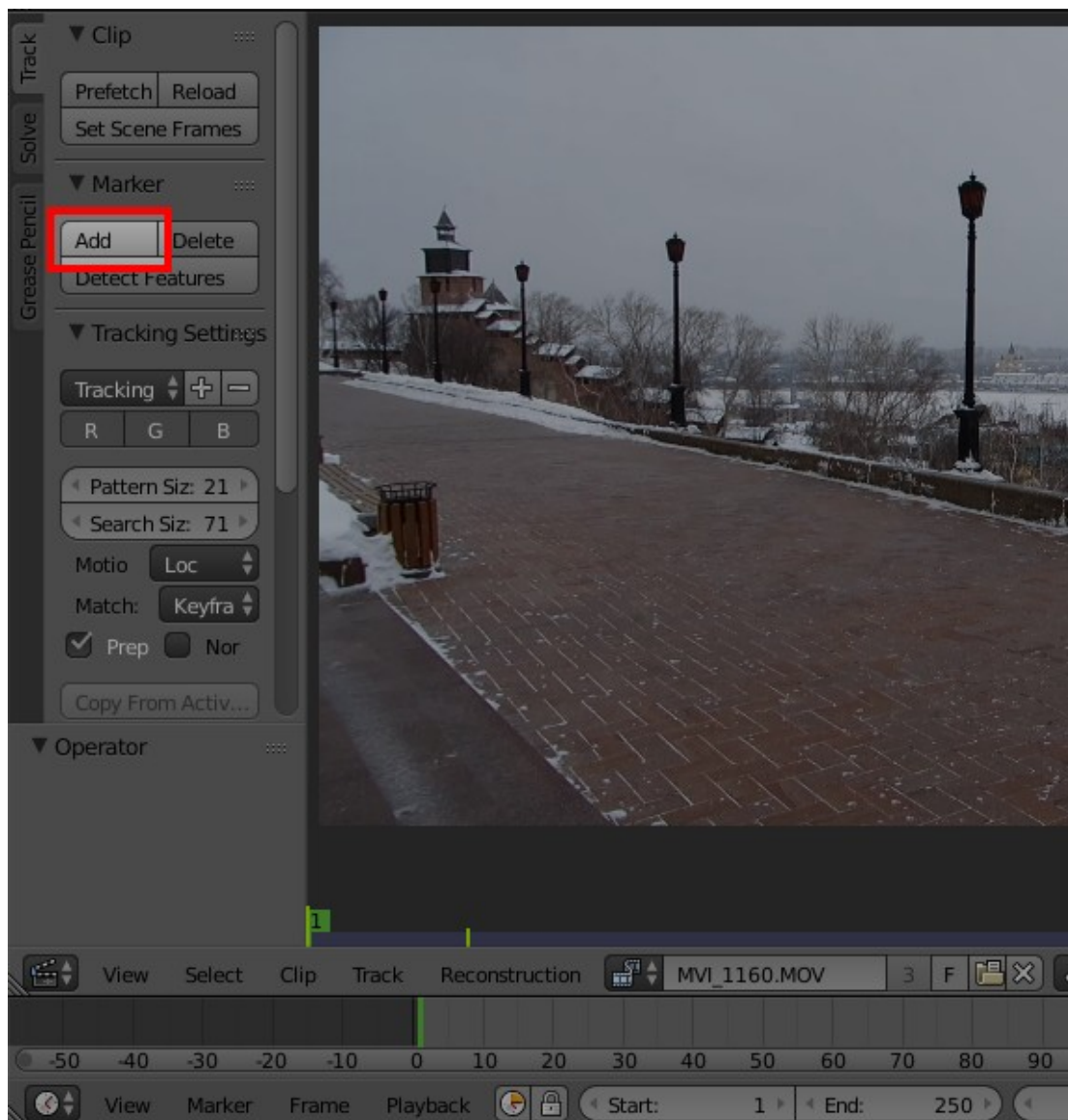
Окно Motion Tracking в Blender'e

Запустим Blender. На момент написания данного пособия актуальной является версия 2.79. После запуска программы перейдём в режим «Motion Tracking»:



Появилось больше количество окон. Сразу откроем видео (Alt+O или кнопка Open).

Для добавления маркера нужно нажать кнопку Add или — что проще и быстрее — зажав Ctrl, ткнуть курсором в нужное место.



Как устроен маркер

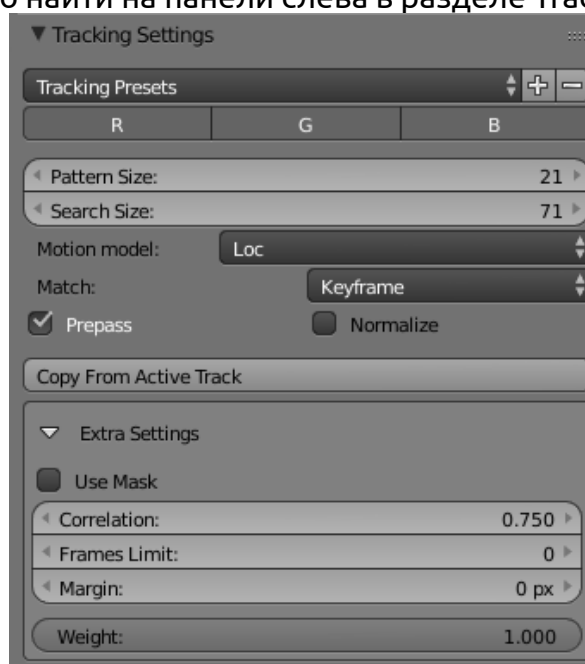
Любой маркер включает 2 основные части — т. н. «**pattern**», то есть часть изображения, **которую** Blender будет пытаться отыскать на следующем кадре, и область **поиска** — область, **в которой** Blender в следующем кадре будет искать pattern.



Двигая квадраты в углах можно менять форму и размер pattern'a. Область поиска можно изменять в размере и перемещать (если известно, что камера движется только влево, например, то область поиска можно сместить вправо, чтобы Blender не пытался искать pattern там, где его быть не может).

Свойства маркера

Настройки маркеров можно найти на панели слева в разделе Tracking Settings.



R, G, B - в каком канале будет отслеживать изменения Blender. В ряде случаев выбранная нами точка более отчётливо видна, скажем, в синем или зелёном канале, нежели в RGB. Можно поэкспериментировать и посмотреть, где оно лучше выглядит.

Pattern Size, Search Size - думаю, объяснять не требуется.

Motion Model - какого рода изменения выбранных точек будет пытаться отслеживать Blender. По умолчанию выбран вариант Loc, то есть Blender будет отслеживать только

изменение местонахождения. В большинстве случаев этого достаточно. LocScale - перемещение и изменение размеров, LocRot - перемещение и вращение, LocRotScale - всё вышеперечисленное, Perspective - изменения перспективы, Affine - загадочная опция, внятных объяснений по которой найти не удалось (разве что «search for markers that are affine-deformed (t, r, k, and skew) between frames»¹).

Match. По умолчанию - Keyframe - это означает, что Blender считает первый кадр эталонным и каждый новый кадр сравнивает с этим эталоном. Если маркер в какой-то момент соскакивает с отслеживаемого объекта и вы устанавливаете его заново вручную — момент, где вы это сделали, считается новым эталонным кадром и Blender ориентируется уже на него, а не первый кадр. Обратите внимание на тонкую жёлтую полосу под видео. Видите более яркие жёлтые точки? - это и есть эталонные кадры.

Если выбрать Previous frame - будет, соответственно, сравнивать не с эталонным, а с предыдущим. Последний вариант неплохо подходит для случаев, когда видео не очень чёткое (например, камера быстро двигается) или объект меняет свою перспективу (в случаях, скажем, когда камера перемещается относительно объекта). В различных видеоуроках в этих случаях рекомендуется использовать именно Match: Previous frame, и это работает. В этом случае, правда, возрастает вероятность плавного «сползания» маркера куда-то в сторону от отслеживаемого объекта. Проблема решается (по крайней мере, отчасти) с помощью Frames Limit, о котором чуть позже.

Что будет, если попытаться использовать вместо этого Motion model: Perspective - автором не проверялось.

Иногда использование Previous frame противопоказано: например, если в кадре много похожих друг на друга объектов. Пусть лучше процесс 100500 раз прерывается, чем маркер соскакивает на похожий объект и начинает кормить Blender неверной информацией...

Prepass - по умолчанию опция включена, убедительных доводов в пользу её отключения автору неизвестно.

Normalize. По умолчанию - выключено. Вроде бы эта опция может повысить качество трекинга, однако увеличивает и необходимое для этого время.

Correlation. Насколько pattern в очередном кадре будет похож на тот, с которым сравниваем (то есть эталонный или предыдущий - смотря что выбрано в Match). По умолчанию - 0.750. Если поставить 1 - Blender будет пытаться найти 100% совпадение, что, очевидно, повлечёт fail.

Frames Limit. По умолчанию - 0, значит, лимит не установлен и Blender будет пытаться отслеживать выбранный объект от первого до последнего кадра. Если установить Frames Limit, скажем, на 25 - то Blender будет отслеживать 25 кадров и останавливаться, для отслеживания ещё 25 кадров соответствующую кнопку придётся жать снова. Удобно, когда видео не очень качественное, маркер часто «соскакивает» (об этом ниже) и хочется лучше контролировать процесс.

Margin. Расстояние маркера до границ кадра, на котором трекинг прекращается. Лучше стандартное значение 0 таки поменять. В противном случае может возникнуть ситуация, когда маркер, оказавшись на границе кадра, не выключается, а продолжает там просто висеть, уже ничего не отслеживая и нагло скармливая Blender'у ложную информацию. Как показывает практика, 25-30 вполне достаточно для видео с разрешением 1920×1080.

Стоит также заметить, что настройки маркеров в панели слева распространяются на вновь добавляемые маркеры, но не влияют на уже созданные. Выражаясь юридическим языком — **обратной силы не имеют**. Для изменения параметров уже созданного маркера нужно использовать N-панель справа.

Как добавлять маркеры

При добавлении маркеров стоит руководствоваться следующими правилами:

0. Маркеров должно быть минимум 8 штук. Будет меньше — Blender сразу скажет «ой, всё» «для реконструкции необходимо хотя бы 8 общих треков на обоих ключевых кадрах» («At least 8 common tracks on both keyframes are needed for reconstruction», если выбран

1 URL: http://www.blender.org/api/blender_python_api_2_63_21/bpy.types.MovieTrackingTrack.html

английский язык для всплывающих подсказок). Это, впрочем, не значит, что маркеров должно быть ровно 8. По идее, чем больше — тем лучше. К тому же это даст возможность без лишних раздумий удалить проблемный маркер (а чего их экономить, ежели их тьма тьмущая...). Подробнее о «At least 8 common tracks on both keyframes are needed for reconstruction» - в разделе «Фотограмметрия».

1. Маркеры должны помещаться на неподвижные объекты. Люди, движущиеся машины, птицы и т. п. подходят плохо. Очень плохо.
2. Маркеры должны быть более-менее по всей площади кадра. Не очень хорошо, если имеется лишь небольшая кучка маркеров в углу/центре/ещё где, а большая часть кадра ими не заполнена. Хотя если этой частью является, к примеру, чистое синее небо — то маркеры там ставить особо и негде...
3. Не стоит ставить маркеры на похожие объекты:



Если в области поиска окажется другой объект, похожий на выбранный, то маркер с высокой вероятностью «соскочет» на него.

4. На **переднем** плане лучше разместить **ПОБОЛЬШЕ** маркеров. Иначе будет нечто, что автор одного англоязычного урока назвал «flickering'ом». Что автор имел в виду — я так и не понял (хотя предположения есть), но, очевидно, абсолютно ничего хорошего.
5. Если камера не только дрожит в руках владельца, но и **поворачивается**, маркеры тем более следует размещать на расстоянии друг от друга, дабы Blender лучше «прочувствовал» оси вращения.
6. Плохой вариант размещения маркера — точка, в которой некий объект пересекается с чем-то на фоне.



На первый взгляд, точка довольно контрастна. Однако эта точка «плавающая»: по мере движения камеры расположение границы реки меняется относительно этого столба. Соответственно, точка тоже будет скользить вдоль столба вверх или вниз, давая Блендеру недостоверную информацию.

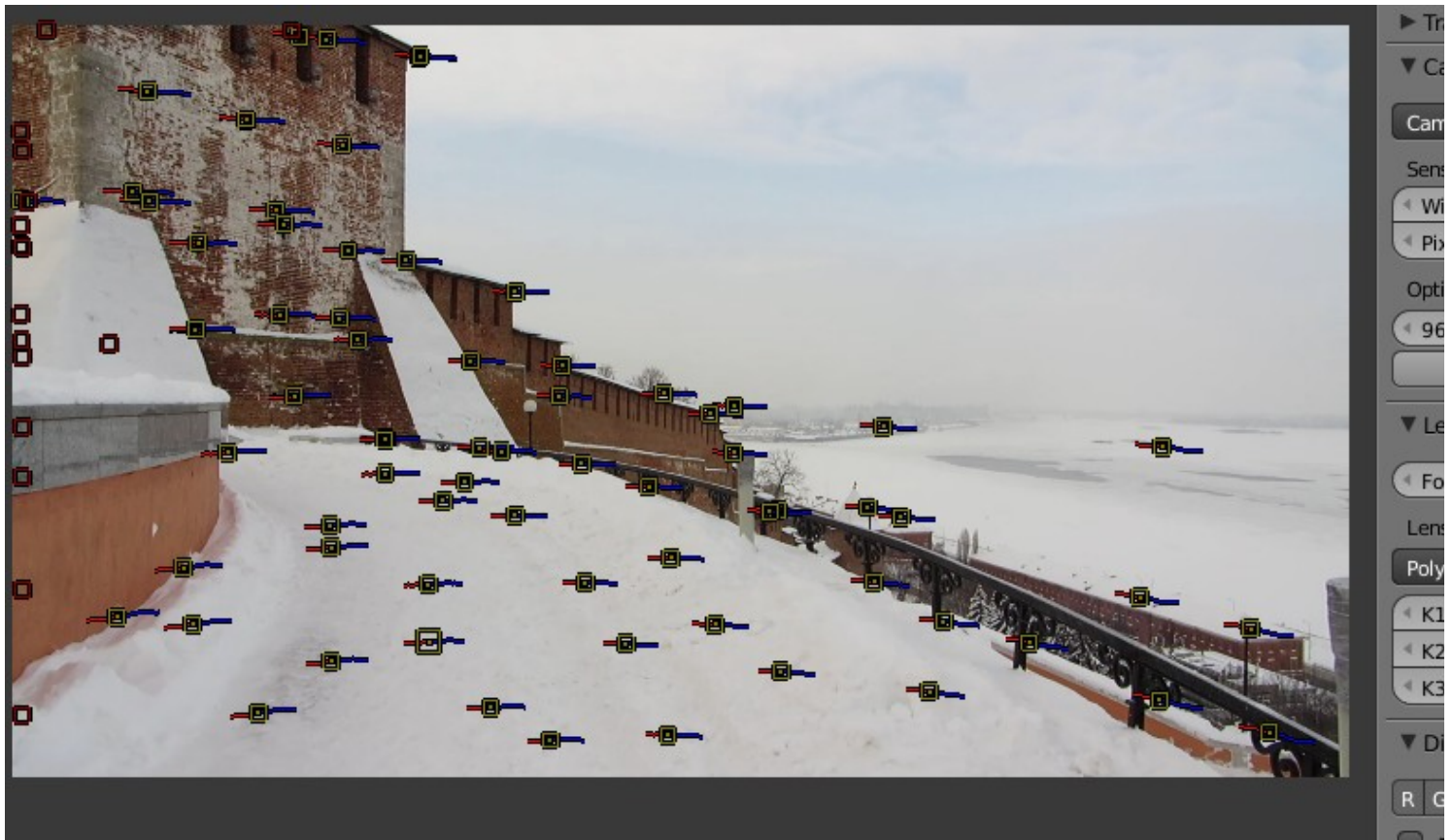
Вручную или автоматически?

Маркеры можно либо расставлять вручную, либо нажать кнопку «**Detect Features**» (панель слева). Она постарается найти более-менее подходящие объекты и установить на них маркеры. Параметр Threshold отвечает за количество маркеров (чем ниже Threshold — тем больше маркеров). Также можно размещать маркеры по всему кадру/в области, очерченной Grease Pencil'ом/за пределами области, очерченной Grease Pencil'ом.

Авторы некоторых видеоуроков считают «Detect Features» бесполезной функцией. Пожалуй, согласиться с этим нельзя, Detect Features — штука довольно полезная, ибо позволяет экономить время. Однако после нажатия соответствующей кнопки следует проверить каждый из добавленных маркеров на предмет соответствия [вышеназванным правилам](#) (ибо автоматике Blender'a они глубоко неинтересны). Например, Detect Features любит ставить маркеры на людей. Но до тех пор, пока автоматическое добавление сразу кучи маркеров с последующим удалением заведомо негодных отнимает меньше времени и сил, чем размещение схожего количество маркеров вручную — использование Detect Features вполне оправданно.

Если камера слишком сильно перемещается в пространстве и/или поворачивается, то, очевидно, Detect Features нажимать придётся несколько раз (ибо маркеры, появившиеся при первом нажатии, вскоре уплывут за пределы кадра).

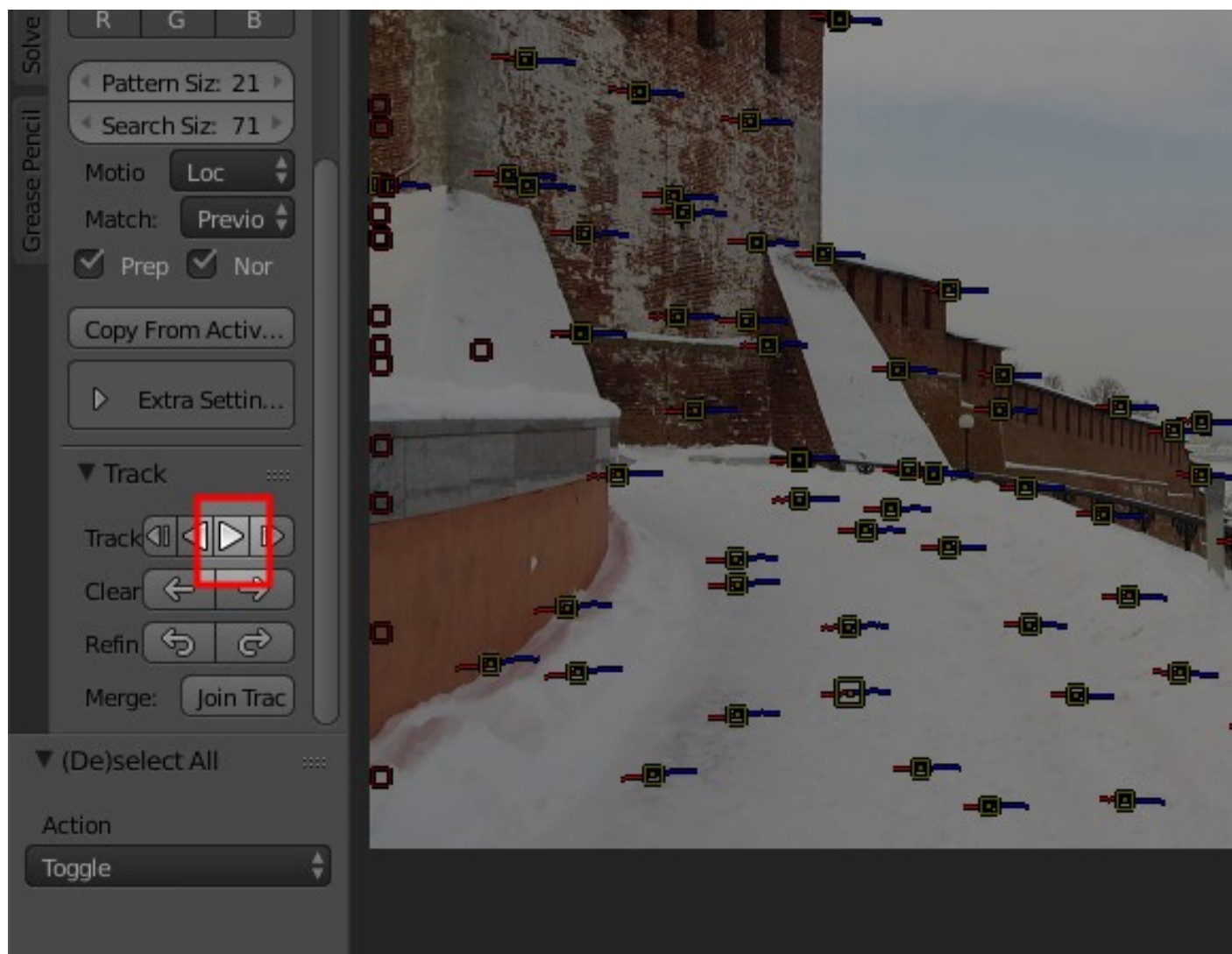
Пример расположения маркеров.



Трекинг

Когда все маркеры расставлены, нужно выполнить, собственно, трекинг. На этом этапе Blender будет отслеживать перемещение каждого маркера (и, соответственно, объектов, на которые эти маркеры установлены) на протяжении всего видео.

Выделяем маркеры (можно выделять по одному, можно — все сразу, нажатием кнопки A). На панели слева находим раздел Track. Что там имеется?



Кнопки ► и ◀ запускают трекинг. Выполнять его можно в любом направлении. Для экономии времени можно поступать следующим образом: ставим курсор на 1-ый кадр, выбираем маркер, жмём кнопку ►, когда Blender дойдёт по последнего кадра — выбираем другой маркер, жмём ◀ и «идём» обратно.

По умолчанию Blender будет пытаться «пройти» всю временную шкалу. Но если у маркеров в параметре Frames Limit (смотри выше) установлено что-то, отличное от нуля, то Blender будет останавливаться каждые n кадров.

► и ◀ выполняют аналогичную операцию, но уже покадрово. Применяется в сложных случаях, когда автоматически Blender отследить перемещения объекта не может. Однако если трекинг срывается — это ещё не повод отслеживать каждый маркер вручную кадр за кадром. Можно попробовать увеличить зону поиска или уменьшить pattern.

Как я понял, Blender смотрит на то, какие точки двигаются быстрее, какие медленнее, таким образом определяя расстояние от камеры до этих точек (медленнее - значит, дальше).

В случае, если отслеживаемая точка находится где-то сзади, её могут регулярно перекрывать другие, более близкие к камере объекты. Как действуем в таких случаях:

0. Отслеживаем точку, пока она не окажется перекрыта неким препятствием. В этом месте процесс трекинга должен остановиться. Иногда вместо остановки маркер соскакивает на другой объект/цепляется за область, в которой объект перекрывается, и начинает следить уже за ней, тоже, очевидно, давая не ту информацию Blender'у. Такие проблемы надо отслеживать и устранять (**Clear** → / ←, см. ниже).
1. Как только объект оказался перекрыт неким препятствием — двигаем курсор на шкале времени (можно стрелками →/← на доске с кнопками) вперёд до того момента, когда объект снова покажется из-за укрытия. Покажется полностью, а не небольшим куском!
2. Отслеживаем объект до следующего препятствия.

3. Goto 0².

Clear → / ← «очищает» трекинг до/после текущей позиции курсора на шкале времени. Используется тогда, когда маркер до определённого момента отслеживался хорошо, после — куда-то соскочил или ещё как-то начал вести себя нехорошо.

Обратим внимание на красные и синие линии, появившиеся на маркерах после трекинга. Они показывают траекторию перемещения маркера. Поскольку перемещение маркера, как подсказывает Капитан Очевидность, зависит от движения камеры, все маркеры, будучи снятые одной камерой, должны двигаться примерно одинаково. Ну и их траектории, таким образом, должны примерно одинаково выглядеть. Если это правило нарушается — значит, у «выделяющихся» маркеров проблемы. И у вас тоже.



И ещё. После успешного трекинга маркер лучше сразу **заблокировать** (выделить и нажать Ctrl+L), чтобы при дальнейшей работе случайно не испортить его.

Фотограмметрия

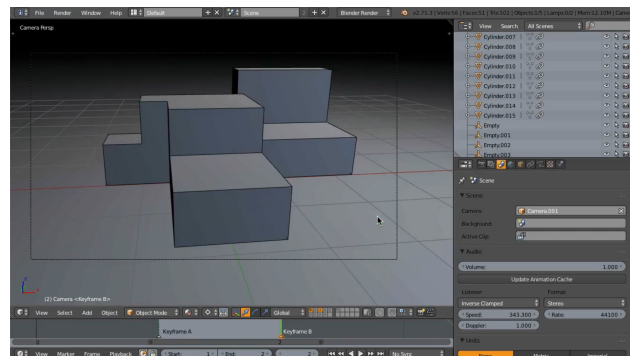
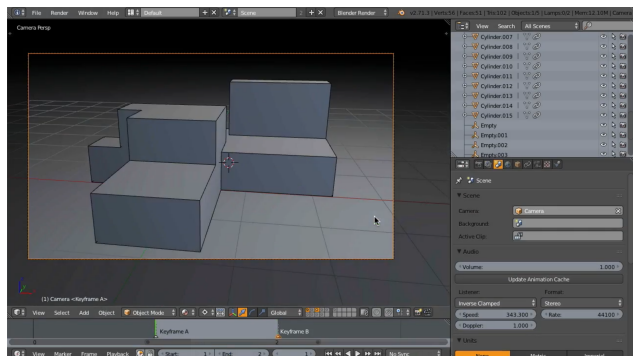
Важная тема, которую нельзя обойти стороной. Долго не мог понять её, но когда понял — хороший трекинг начал получаться практически всегда с первого раза. Понимание этой темы позволит сразу выставлять правильные значения KeyFrame A и KeyFrame B вместо того, чтобы пытаться их выставить наугад. Почему-то в русскоязычных уроках тема не затрагивается.

Взглянем ещё раз на картинку выше. Мы понимаем, что одни маркеры находятся ближе (в паре метров от камеры), другие — дальше (за рекой, например). Blender этого **не** понимает. Для него все маркеры — просто точки на **плоской картинке**, а не в трёхмерном пространстве. Об удалённости каждого маркера от камеры в таких условиях говорить затруднительно, а пока Blender не «поймёт», что ближе, а что дальше, он не сможет рассчитать перспективу и создать нормальную виртуальную камеру. Значит, наша задача — помочь ему «понять». А для этого,

² Подробнее — в уже упомянутом выше «How to do Camera Tracking in Blender» от Oliver Villar.

соответственно, надо кое-что понять самим.

Нам понадобится 2 ключевых кадра, на которых изображено одно и то же, но имеется существенная разница в перспективе. Одна и та же локация. Но от первого до второго кадра камера совершила ощутимый поворот или иное движение, создав разницу в перспективе. Наглядный пример, позаимствованный из курса «Track, Match, Blend! - 2»:



Если присмотреться, станет заметно, что эта сцена состоит всего из двух кадров. **Это — те самые KeyFrame A и KeyFrame B**, что по умолчанию выставлены в 1 и 30 соответственно. А вот пример более приближённый к жизни:



Здесь в Blender загружен полноценный видеоролик, а на картинках выше — кадры 100 и 170. Поскольку эти два кадра отличаются друг от друга достаточно заметно, именно они и должны стать Keyframe A и Keyframe B. Таким образом:

0. Оставлять Keyframe A равным 1, а Keyframe B равным 30 уместно далеко не всегда.
1. Пытаться охватить весь таймлайн (скажем, если в сцене 300 кадров — установить Keyframe A = 1, Keyframe B = 300) тоже обычно не никакой необходимости.

Что происходит после нажатия кнопки «Solve Camera Motion»?

0. Blender смотрит, какие маркеры есть на Keyframe A.
1. Blender находит эти маркеры на Keyframe B.
2. Blender анализирует, как положение одних и тех же маркеров изменилось на Keyframe B по сравнению с Keyframe A. На основе этих данных Blender определяет перспективу.

Важно! Должно быть как минимум 8 маркеров, которые имеются и на Keyframe A, и на Keyframe B! Если не соблюсти это правило — получим ту самую ошибку «**At least 8 common tracks on both keyframes are needed for reconstruction**». То есть получив такое сообщение от Блендера, нужно добавлять маркеры не наугад по принципу «лишь бы ПОБОЛЬШЕ!», а так, чтоб на кадрах А и Б было от 8 и более общих маркеров.

Ещё одна возможная причина возникновения подобной ошибки — кадра А или кадра Б просто не существует. Пример: сцена начинается с кадра 20, а заканчивается кадром 150. KeyFrame A == 1, KeyFrame B == 30. С кадром В всё понятно, а вот кадр А явно выходит за рамки

таймлайна (если сцена начинается с 20-го кадра, где Блендеру искать первый?).

Для чего нужна опция Tripod?

На случай, если исходное видео снималось со штатива, причём с панорамной головкой.

Как уже сказано выше, Blender после нажатия Solve Camera Motion пытается вычислить перспективу. Однако при съёмке со штатива, да ещё и с панорамной головкой, перспективы нет, паралакса тоже. Следовательно, традиционный способ работы не подходит. На этот случай был введён особый алгоритм.

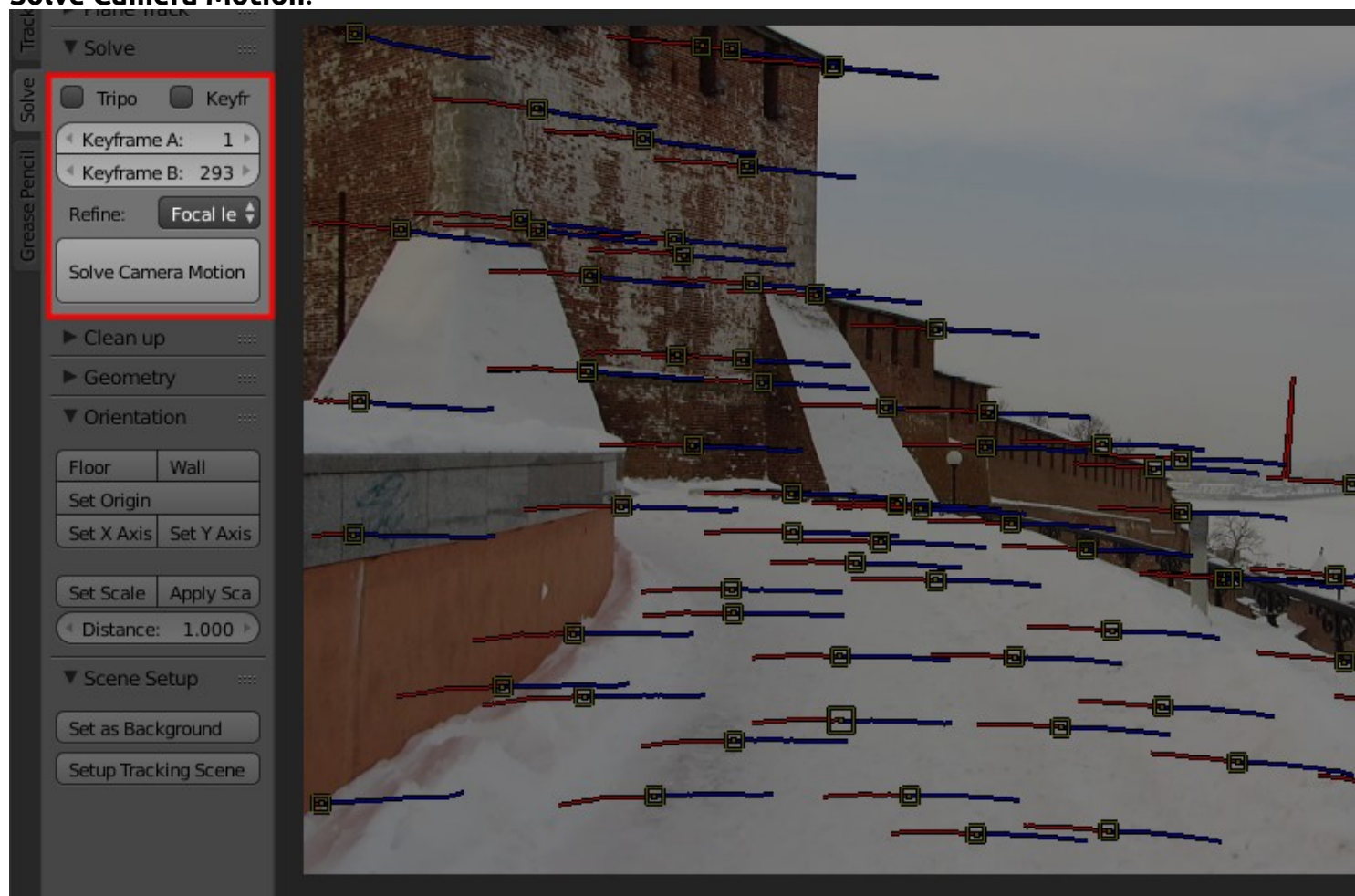
Иногда опция Tripod хорошо себя показывает и без штатива - скажем, если оператор стоял на одном месте и почти не двигал камерой (только поворачивал). Но не всегда.

Для чего нужна опция Keyframe?

Всплывающая подсказка гласит: автоматически выбирать ключевые кадры при расчёте движения камеры или объекта. В общем, те, кто так и не понял, что за Keyframe A/B и не знает, какие значения в них указывать, могут попробовать эту опцию. На практике она вряд ли даст хороший результат, так что лучше установку Keyframe A/B автоматике не доверять.

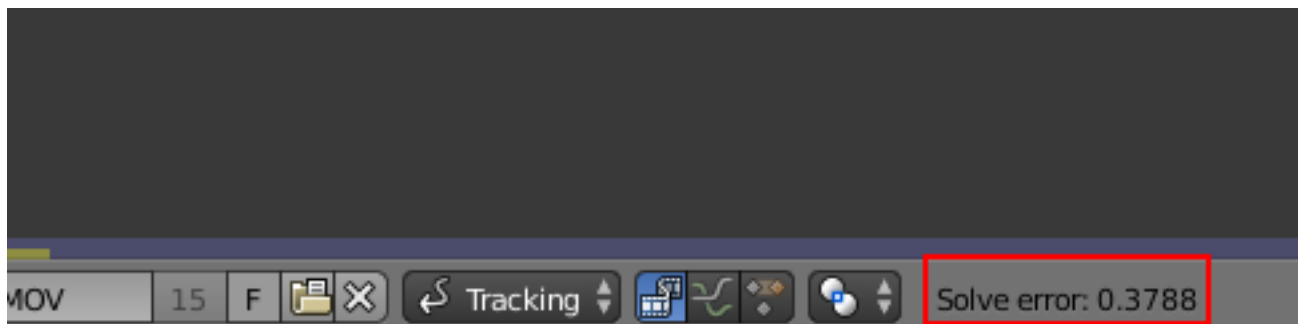
Создание виртуальной камеры

Когда трекинг выполнен, на панели слева переходим во вкладку Solve и нажимаем кнопку **Solve Camera Motion**.



Параметры Keyframe A и Keyframe B можно пока оставить стандартными (то есть 1 и 30 соответственно). Не будем пока трогать также Tripod, Keyframe, Refine. Просто жмём Solve Camera Motion и смотрим, что получится.

Если же всё прошло успешно, то внизу отобразится текст «**Solve error**» и некое значение.



Какие значения считаются идеальными, какими — приемлемыми, а на каких весь проект подлежит незамедлительной отправке в /dev/null — вопрос спорный. Более-менее общепризнанно, что значение меньше 0.5 — идеальный результат, 0.5-1 — очень хороший. Приемлемым кто-то называет диапазон 1-2, кто-то — 2-3, всё, что выше — никуда не годиться. На практике лучше самому посмотреть, как выглядит результат в окне 3D-вида и оценить, насколько он приемлем.

Информация о камере

Эти параметры можно найти в документации к камере. По крайней мере, фокусное расстояние и размер матрицы. Если информация недоступна или доступна не полностью — можно попробовать подобрать их самому. Например, для своей камеры я не нашёл значений K1 — K3. Их можно попробовать **подобрать вручную**: увеличиваем/уменьшаем значение, жмём Solve Camera Motion (Shift+S) и смотрим, как это повлияет на Solve error. Если стал меньше — значит, продолжаем изменять в том же направлении значение параметра (если уменьшали — делаем ещё чуть меньше, если увеличивали — соответственно, чуть больше) и снова смотрим на результат. Так до достижения наименьшего значения Solve error'a.

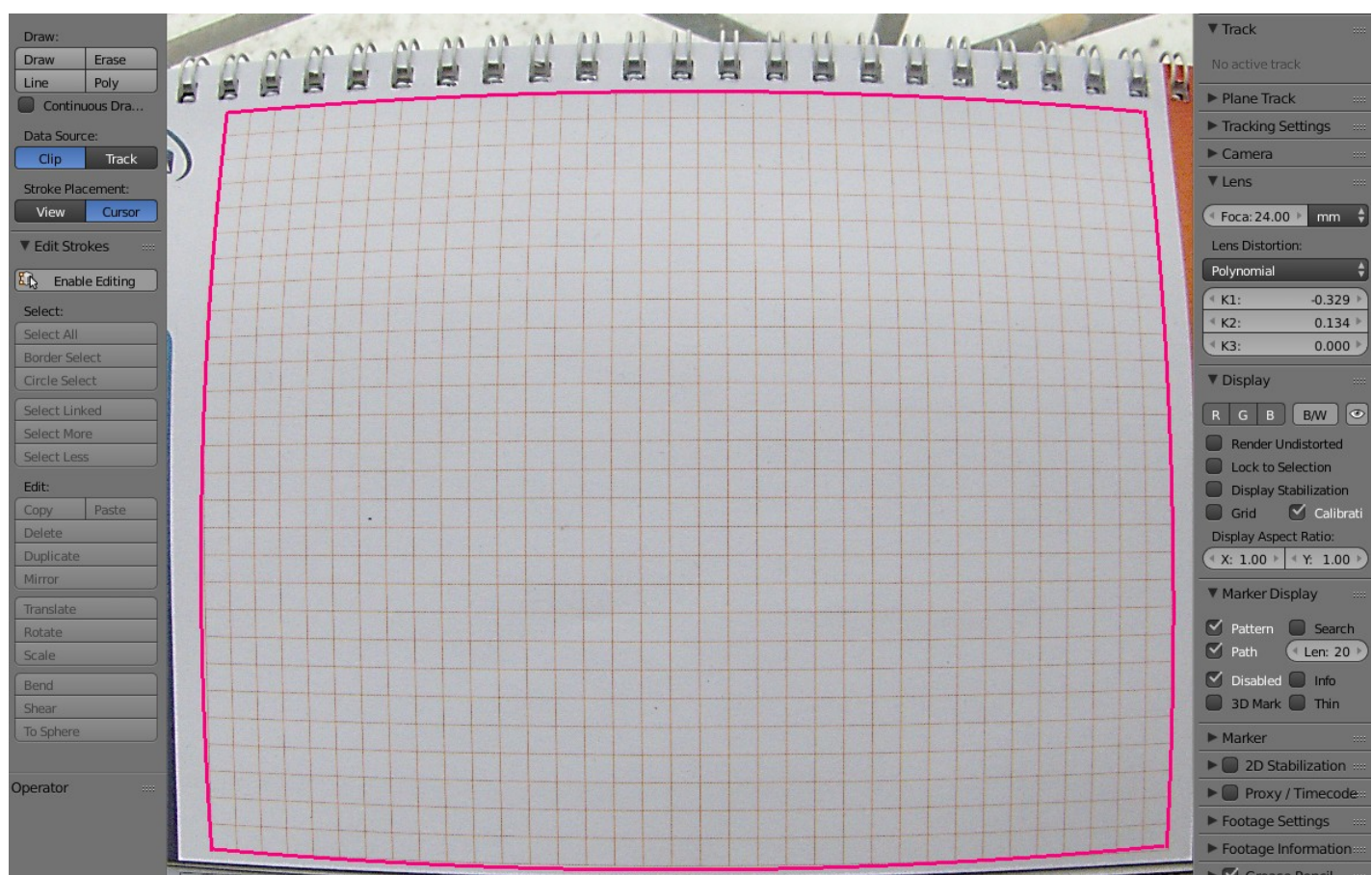
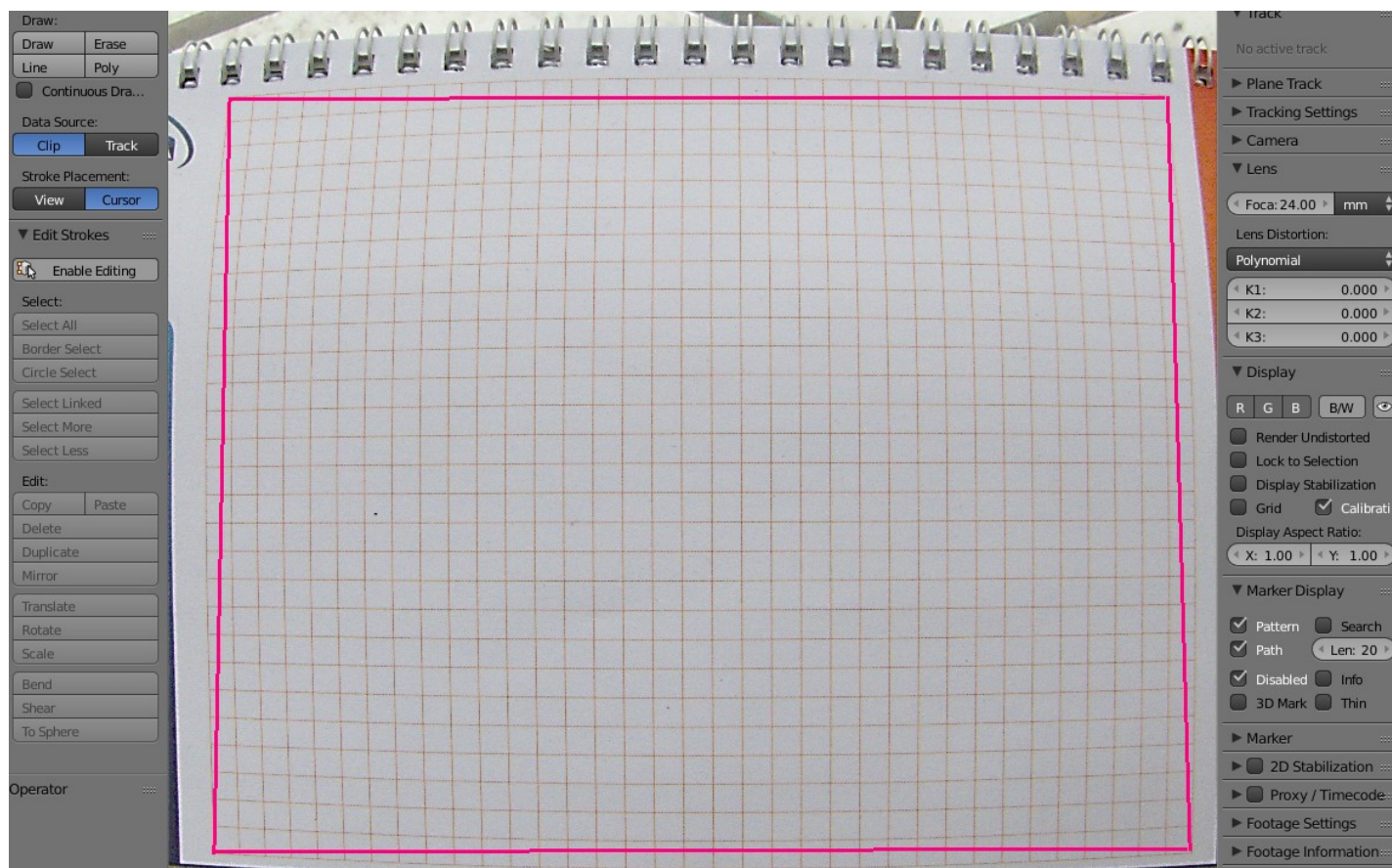
Примечание. Как показывает практика, K1 используется довольно часто, K2 — реже, K3 почти никогда трогать не требуется.

Если гадать не хочется, можно попробовать подобрать значения K1-K3 иным путём. Для определения параметров конкретных линз можно сделать снимок чего-то, что хорошо показывает искажения оптики (например, сетка, или что-то ещё, где есть прямые линии). Тетрадный лист в клетку подойдёт.

Включаем Grease pencil и рисуем прямые линии поверх сетки (или что там ещё было снято). Эти нарисованные Grease pencil'ом линии будут выступать в качестве эталона.

В правой панели (N) в разделе «Display» включаем опцию «**Calibration**». На той же панели находим и включаем «**Render Undistorted**».

Теперь можно экспериментировать с параметрами K1-K2 и смотреть на результат. По мере изменения этих параметров будут изгибаться нарисованные Grease pencil'ом линии (при условии, что опция «Calibration» включена!). Менять эти параметры следует до тех пор, пока линии Grease pencil'a не будут более-менее совпадать по своим изгибам с линиями на фото (которые искажены оптикой).



Для использования неискажённого изображения в композиторе, следует добавить нод **Movie Undistort** (с включенной опцией undistort) после нода Video Clip.

Другой путь — ничего самому не трогать и предоставить Blender'у возможность попробовать подобрать эти значения автоматически (Refine). В этом случае в ноду Movie Undistort следует использовать опцию «Distort».

Ad notam: Undistort на данный момент имеет 2 варианта:

0. «**Dividions**» — даёт возможность работать лишь с двумя параметрами: K1 и K2.

1. «**Polynomial**» — позволяет работать ещё и с K3.

Основное преимущество варианта «division» в том, что он требует меньше условий, нежели вариант «polynomial» в случаях с сильным искажением. Таким образом, вариант «division» будет более уместным для широкоугольных линз³.

Постараюсь обобщить. **Работа с инфой о камере выглядит примерно следующим образом:**

0. устанавливаем **размер сенсора**. В моём случае (да и со многими зеркальными камерами) это 22.3;
1. устанавливаем **фокусное расстояние**. Тут, пожалуй, лучше не пытаться вспоминать фокусное, при котором снимал видео, а просто установить его на **20**.
2. жмём «**Solve Camera Motion**». Если трекинг выполнен качественно и Keyframe A/B установлены грамотно (см. выше) — вероятно, Solve Error с первого раза выйдет меньше 1, что очень хорошо.
3. Можно попробовать включить опцию **Refine**. Focal Length, K1, K2. Нередко это позволяет снизить Solve Error ещё больше (иногда, впрочем, эффект обратный).

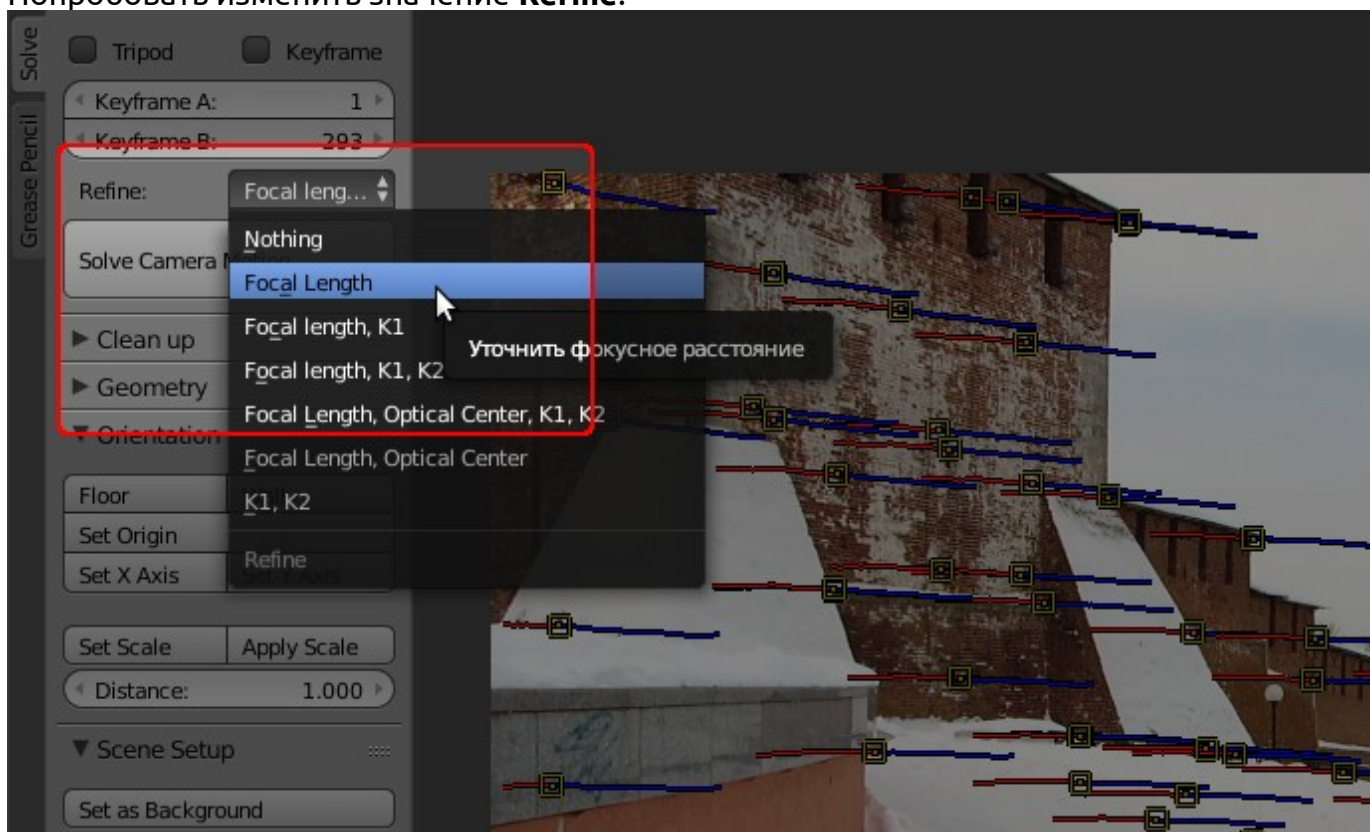
Что делать, если solve error **слишком высокий**?

0. Указать правильную **информацию о камере**, которой снималось оригинальное видео (см. выше).

3 «The main advantage of the division model is the requirement of fewer terms than the polynomial model for the case of severe distortion. Therefore, the division model seems to be more adequate for wide-angle lenses» // URL: <http://blender.stackexchange.com/questions/15620/how-to-tweak-k1-k2-and-k3-undistortion-values-in-motion-tracking>

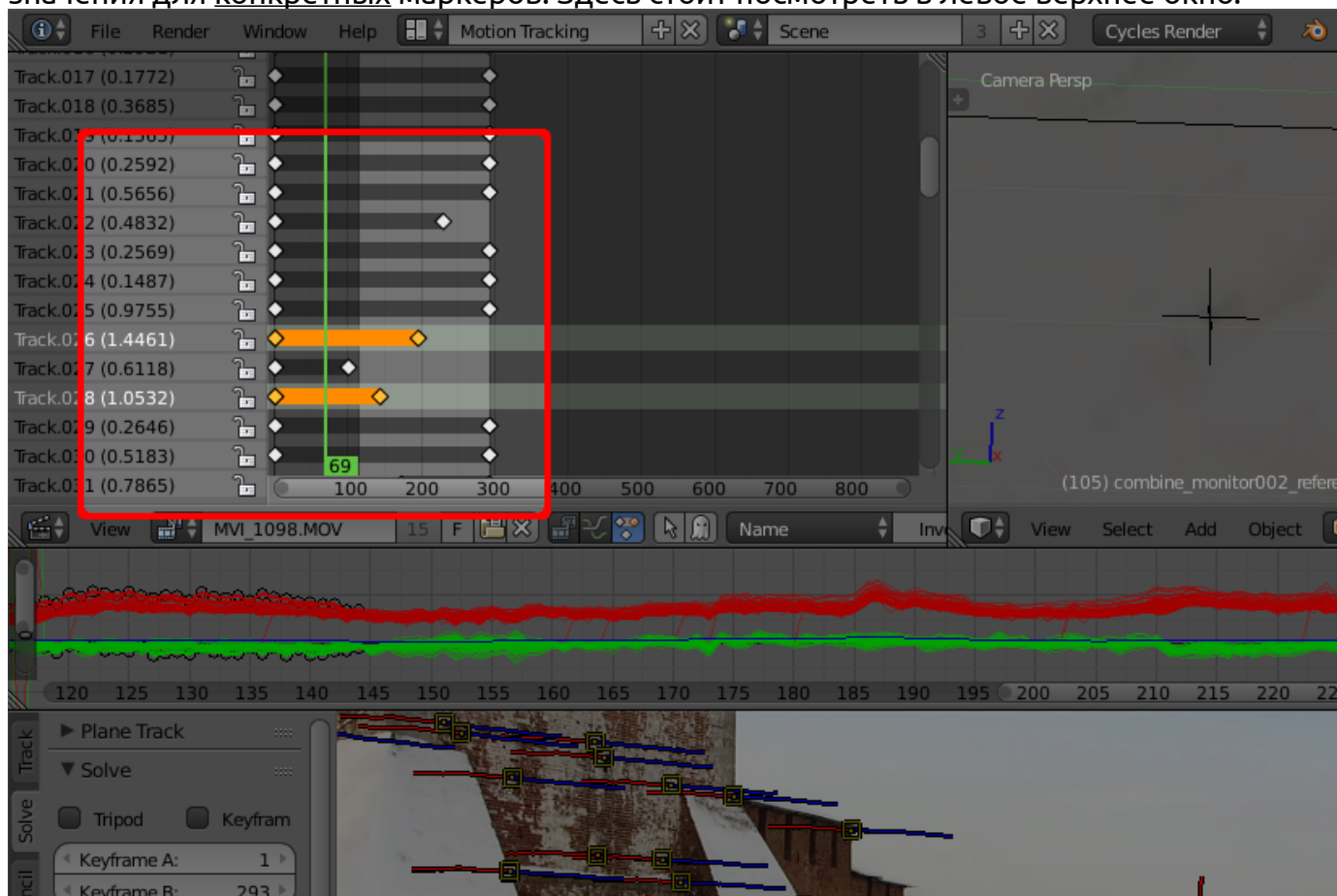


1. Попробовать изменить значение **Refine**.



- В первую очередь — менять Focal Length. Если не помогает - попробовать Focal Length + K1, затем Focal Length + K1, K2 и т. д.
2. Solve error имеет свой собственный у каждого маркера. То, что мы видим внизу, есть средняя величина. Что влияет на среднее значение для всех маркеров? Конкретные

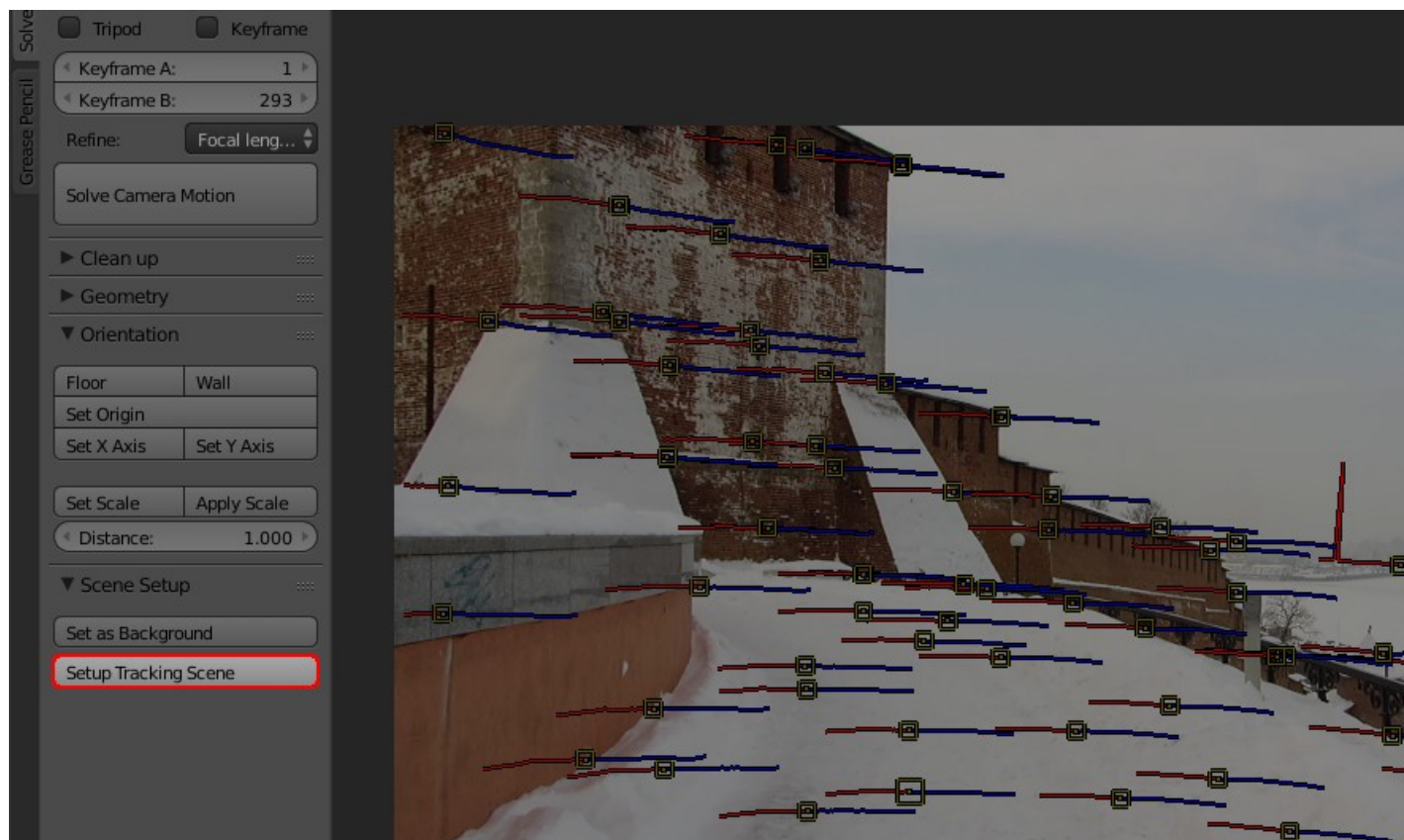
значения для конкретных маркеров. Здесь стоит посмотреть в левое верхнее окно.



В нём отображается список всех маркеров с их solve error'ами. И здесь наглядно видно, какие из них имеют нормальный (меньше 1) solve error, а у каких оных достигает непотребных величин. Здесь же можно выбрать и удалить проблемные маркеры (особенно если их было добавлено много и удаление нескольких результат не испортит). Кстати, здесь же можно выполнить **сортировку** по значению Solve Error (кнопка Name на картинке выше).

Внимание! Удалять имеет смысл лишь те маркеры, которые имеют неприлично высокий Solve Error **при низком** Solve Error других маркеров. Если Solve Error высокий у всех или почти у всех — проблема, вероятнее всего, не в конкретном маркере, а в информации о камере.

Когда средний solve error достигнет приемлемых значений, можно настраивать сцену. Автоматически это может сделать кнопка «**Setup Tracking Scene**».



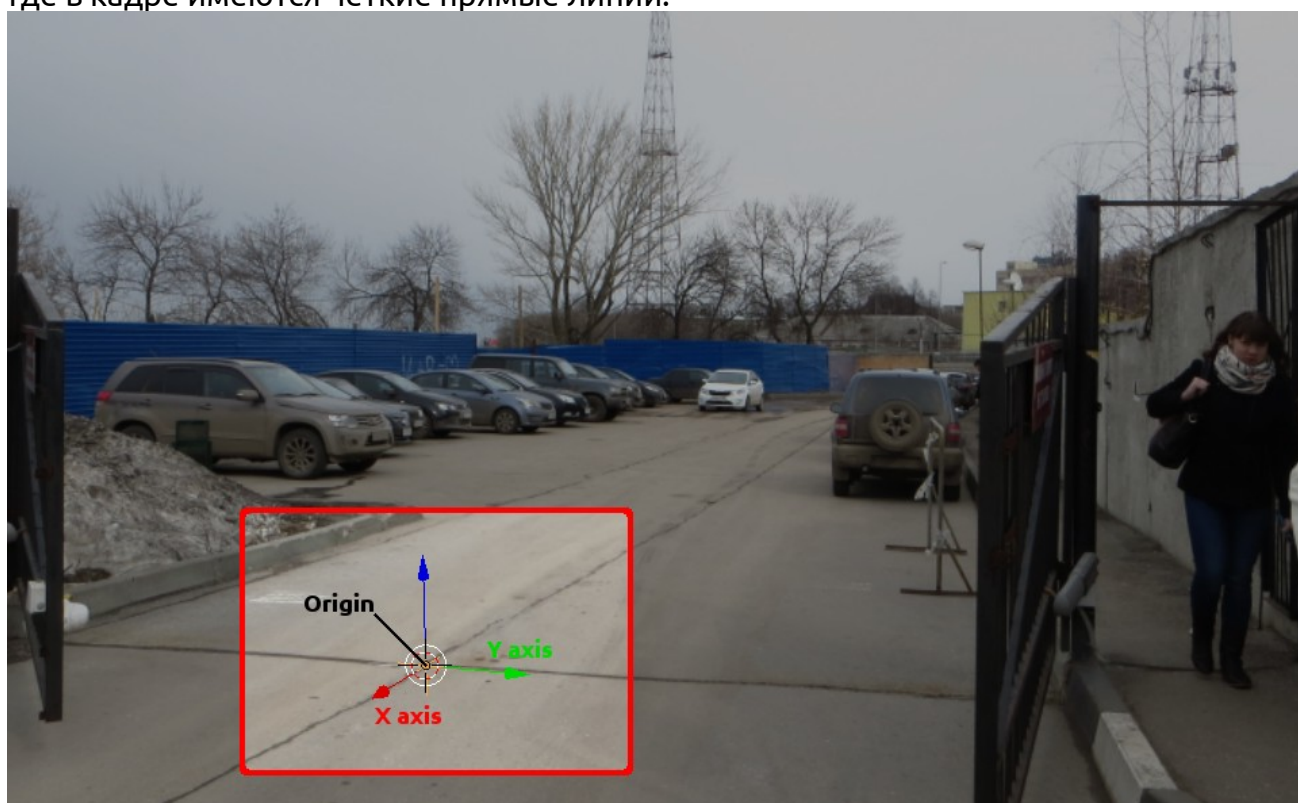
Если конкретно — в качестве фона при виде из камеры ставится видео, с которым мы работаем, камере добавляется констрейнт Camera Solver и т. д.

SetFloor, SetOrigin, SetX/Y

Трекинг выполнен, CameraSolver установлен, но где верх, где низ — Blender по-прежнему не знает. Поэтому надо указать ему, где пол. Для этого выбираем 3 «половых» маркера и жмём кнопку **«Floor»** (она в разделе Orientation).

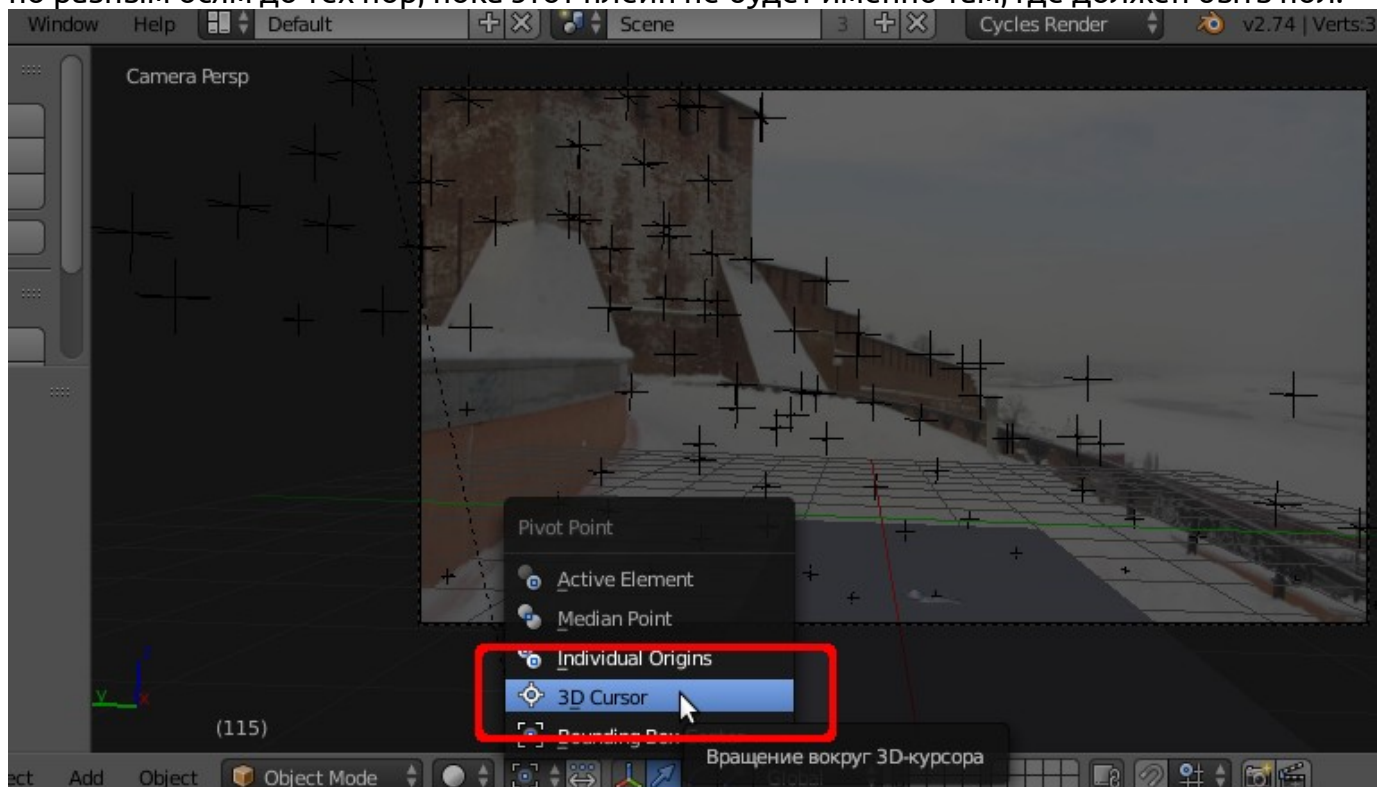
SetOrigin — как я понял, определяет, какой маркер будет «центром мира».

SetXAxis, SetYAxis — определяет, соответственно, ось X или ось Y. Особенно удобно это в сценах, где в кадре имеются чёткие прямые линии:



После нажатия «Floor» «пол» по-прежнему будет непойми где. /* **update:** если Keyframe A/B выставлены грамотно, возможно, пол с самого начала будет более-менее в том месте, где надо */ Поэтому действуем следующим образом:

0. Выбираем 3 «половых» маркера, жмём Floor (в более старых версиях Blender'a эта кнопка называлась SetFloor).
1. Выбираем «центральный» маркер, жмём SetOrigin.
2. Выбираем какой-то маркер, который будет отмечать ось X или Y — смотря что удобнее в конкретной ситуации. Ещё раз: центром координат (и началом вектора) является маркер, который выбран как Origin, а второй маркер, который выбран как ось X/Y — конец вектора.
3. В окне 3D-вида добавляем плейн, который будет полом, выбираем вращение вокруг 3D-курсора (он после нажатия кнопки SetOrigin должен совпадать с соответствующим маркером) и крутим камеру (не плейн, который решили сделать полом, а именно камеру!) по разным осям до тех пор, пока этот плейн не будет именно там, где должен быть пол.

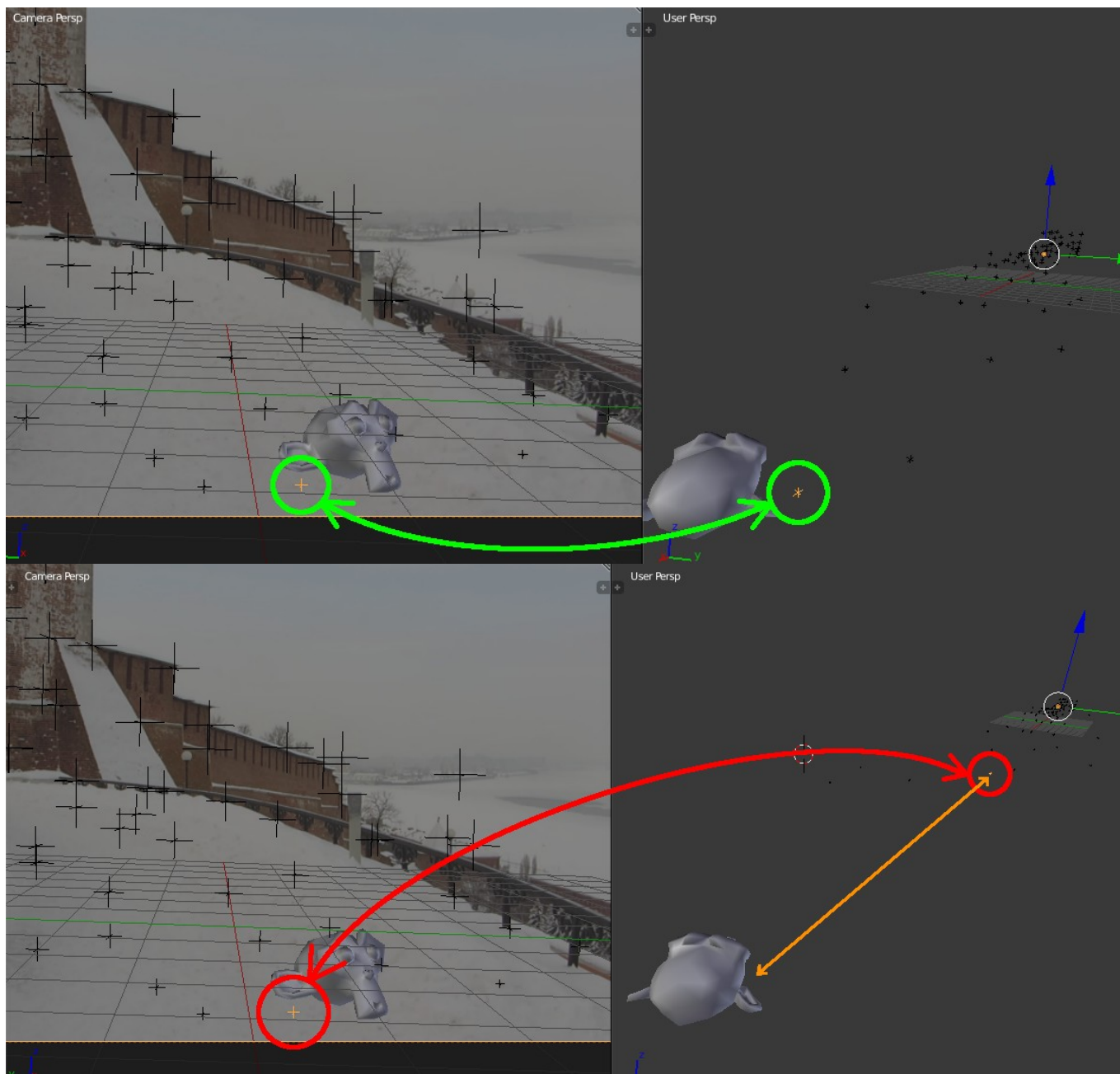


Проверка

Смотрим, что получилось (Alt+A в окне 3D-вида). Если видимых проблем нет, можно отрендерить сцену (для начала с низким разрешением) и посмотреть, что получилось. Однако нередко на этом этапе приходится сталкиваться с рядом проблем.

Проблема 0: объекты «съезжают» в сторону. То есть вроде бы все добавленные в сцену 3D-объекты на своих местах, но со временем слегка (иногда более, иногда менее заметно) «уплывают» в сторону (явление, в англоязычных уроках получившее название «sliding»). Что делать?

0. Ну, для начала — проверить ещё раз маркеры, особенно те, которые находятся рядом со съезжающим объектом. Все ли маркеры стабильно сохраняют своё положение по мере проигрывания анимации?
1. Обратит внимание на то, где расположены добавленные объекты относительно маркеров. Они должны быть примерно в одном месте в окне 3D-вида. Смотреть нужно не из камеры, а именно со стороны.

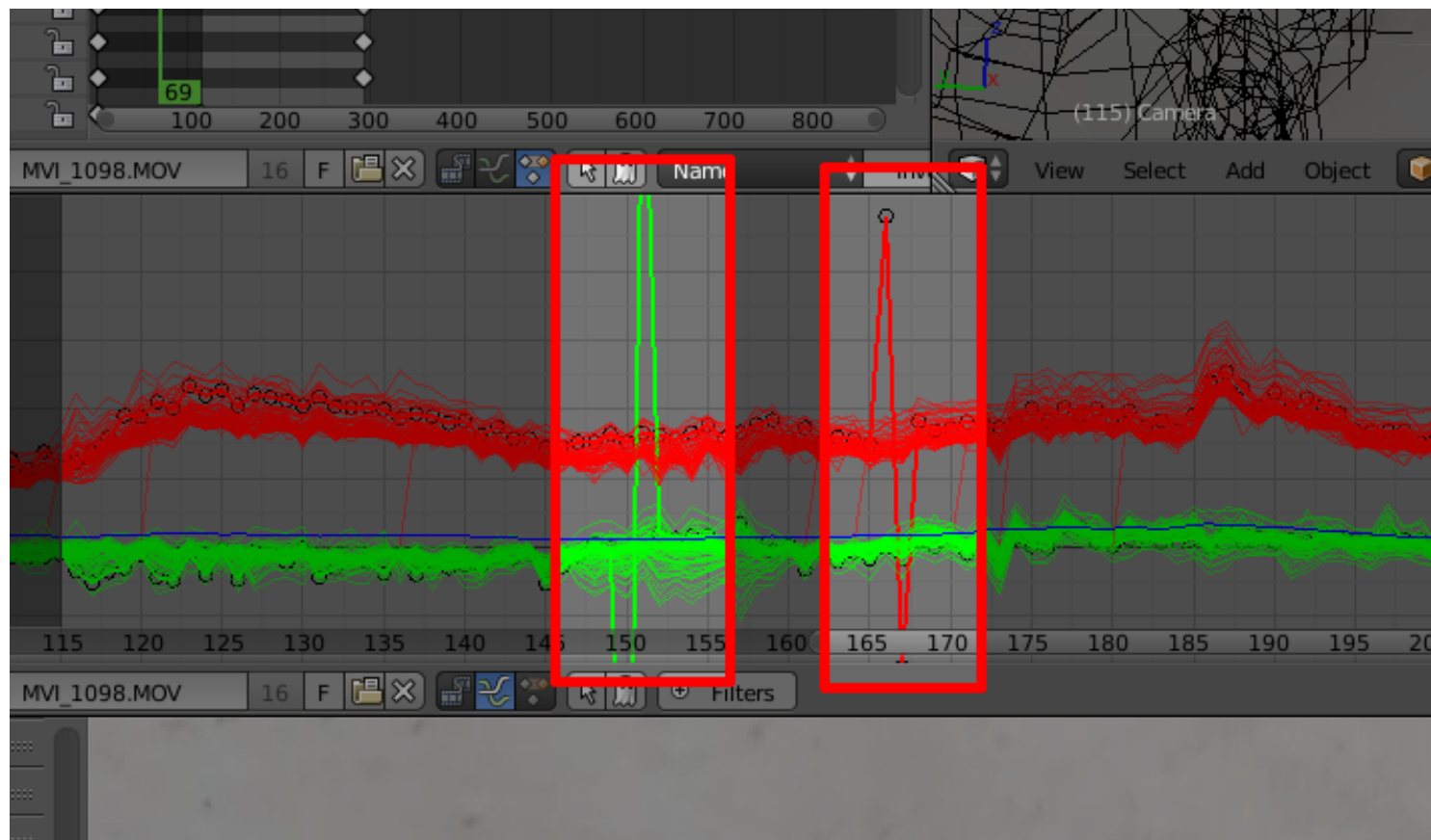


Располагать объект следует таким образом: выделить маркер в окне 3D-вида, нажать Shift+S, выбрать Cursor To Selected для установки 3D-курсора точно на данный маркер, после чего выбрать сам объект, снова нажать Shift+S и выбрать Selected To Cursor. Дальше менять размер или другие параметры по ситуации. И да, по личным наблюдениям — очень часто причина проблемы со «sliding'ом» именно в этом. Хотя в уроках на просторах Интернета об этом, кажется, не рассказывается...

2. Наконец, некоторые видеоуроки советуют [указать информацию о камере](#).

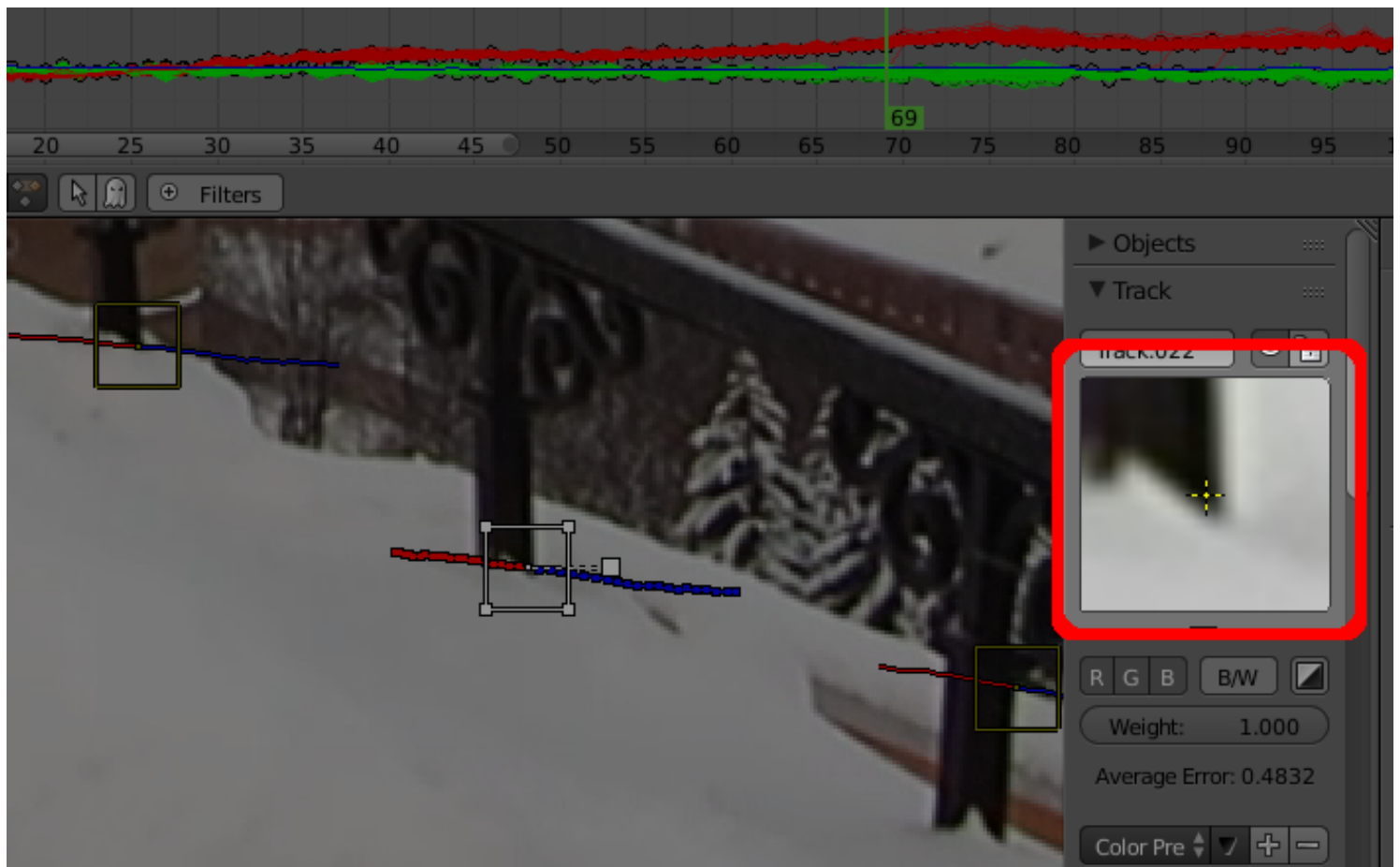
Проблема 1: solve error низкий, вроде всё нормально, но в какой-то момент добавленные объекты (вариант — какой-то один объект) ВНЕЗАПНО прыгает, дёргается и т. п.

Причина, вероятнее всего, в том, что дёргается какой-то из маркеров (даже если ранее оно замечено не было). Ещё раз взглянем на [сюда](#). Вроде бы смотрели траектории маркеров — не нашли ничего подобного... Теперь посмотрим в окно графов (красно-зелёные графики в окне Motion Tracking) —



Явление, которое в англоязычных уроках называется «спайками» («spikes»; как это адекватно перевести на русский — я не придумал, так что пусть будут спайками). Стоит отметить, что «спайками» считаются и являются проблемами только одиночные скачки вверх или вниз у конкретных маркеров. Если траектория каждого маркера обозначается примерно совпадающими линиями — какими бы ломаными они не были — скорее всего, всё в порядке (а если камера сильно трясётся — то и вовсе удивляться нечему).

Здесь хотелось бы сделать важное замечание: спайки бывают **внешними** и **внутренними**. В первом случае мы видим «выпрыгивание» вверх или вниз, продемонстрированное на выше. Однако бывают случаи, когда «наружу» никто не выпрыгивает, но «впрыгивает» «внутрь» (на рисунке этого изобразить не получается, ибо график в этом случае не будет ничем отличаться внешне от нормальных).

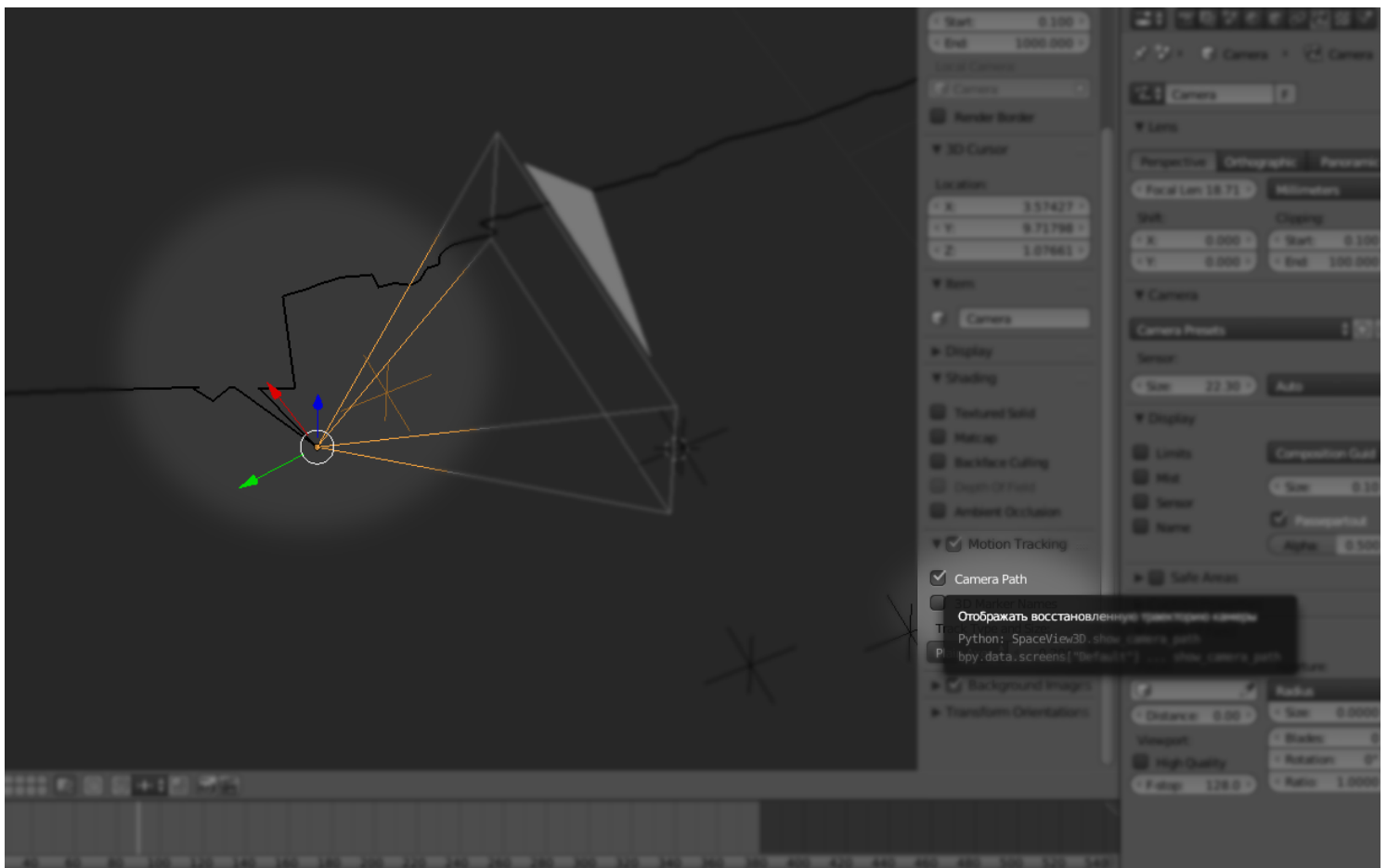


Так что делать? Для начала — определиться, на каком кадре возникают проблемы. Если оные обнаружены уже при просмотре отрендеренного видео в медиапроигрывателе, то можно примерно определить кадр, посмотрев, на какой секунде видео возникла проблема и умножив это число на значение FPS (оное устанавливается в настройках рендера, по умолчанию — 24). Как вариант — OpenGL Render Animation (чтоб быстро отрендерить), включить в разделе Metadata настроек рендера включить **Stamp Output**, ниже отметить чекбокс **Frame** - в этом случае на видео будет отображаться текущий кадр.

Далее — возвращаемся в окно Motion Tracking'a и проверяем **каждый** маркер по очереди — то есть проигрываем (Alt+A) видео в «проблемном» диапазоне (напомню — оный мы определяем приблизительно) и каждый раз внимательно смотрим окно просмотра этого маркера:

Кстати, об этом тоже не удалось услышать ни в одном видеоуроке. Не знаю, почему...

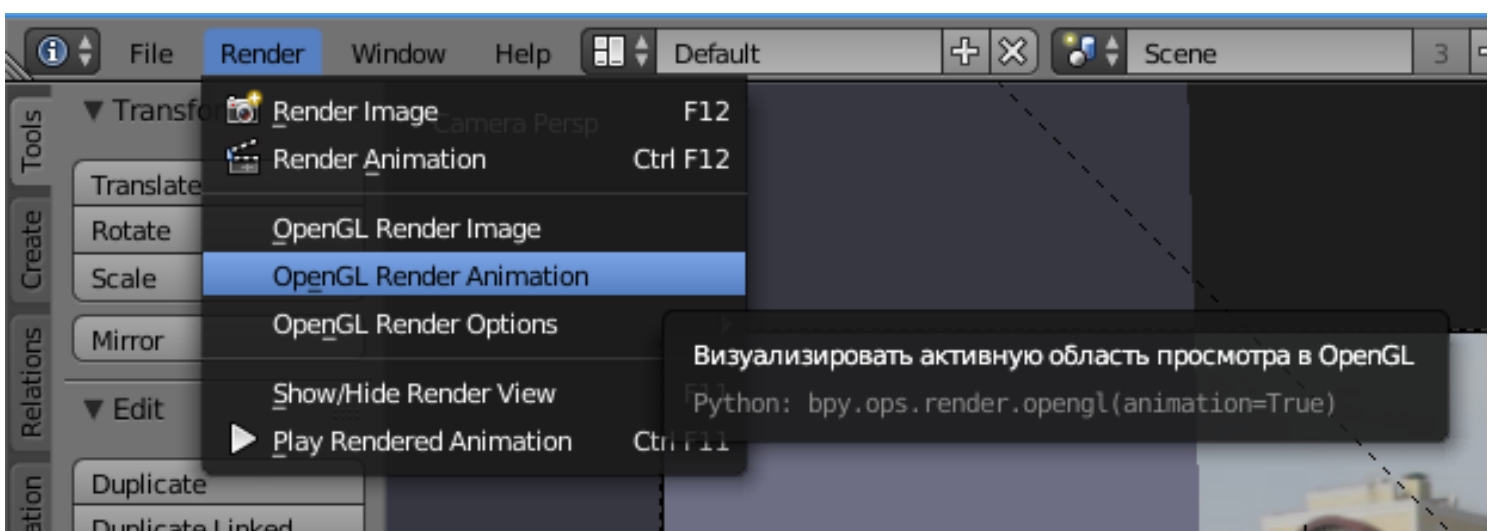
Есть ещё один способ диагностики таких проблем. В окне 3D-вида на N-панели надо **включить чекбокс Camera Path**. Появится чёрная линия — график движения камеры. Резкие рывки виртуальной камеры на нём заметны в виде всё тех же «спайков»:



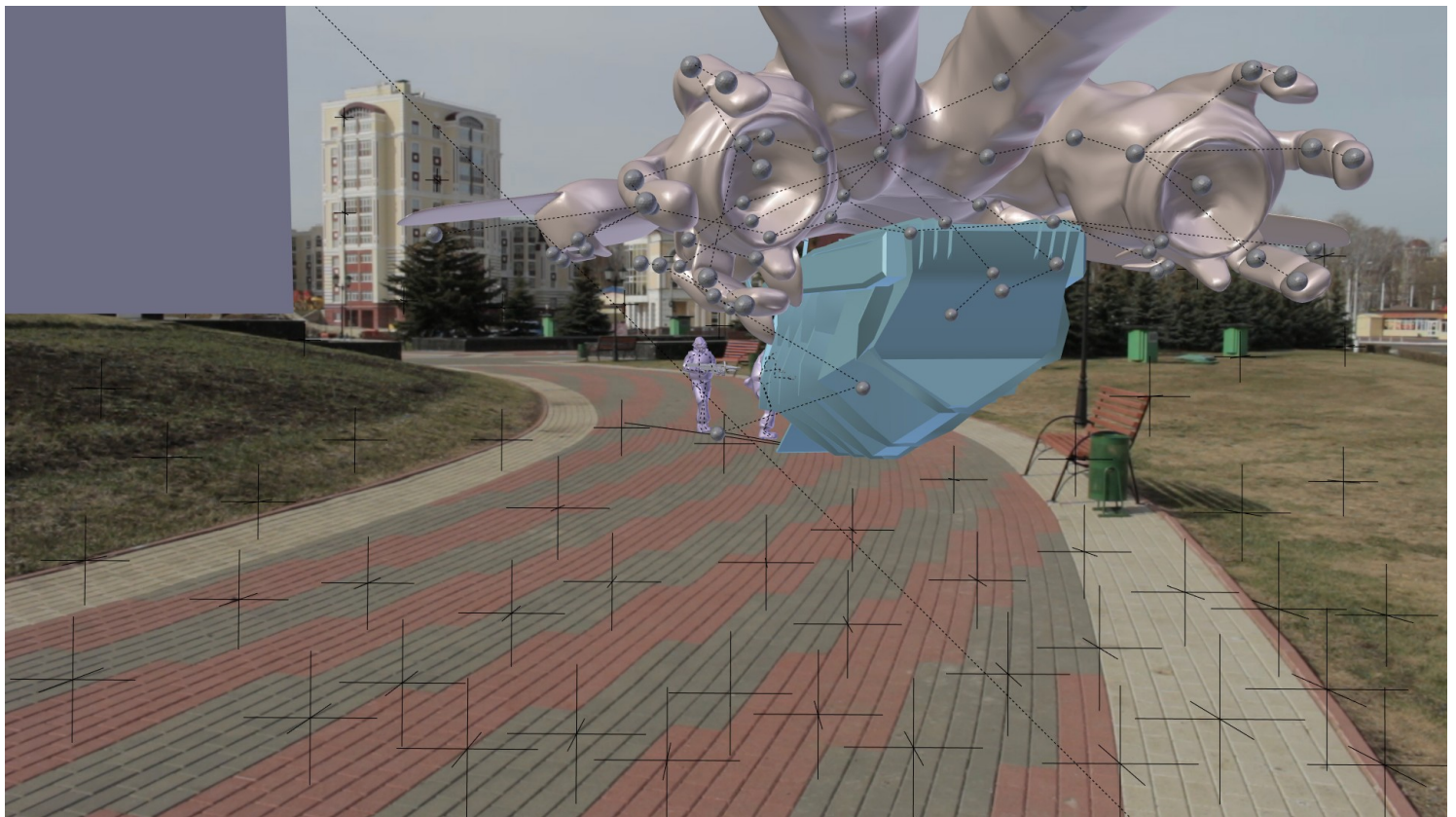
Ad notam: для удобства проверки маркеров можно отключить отображение видео в Movie Clip Editor'e — **Mute footage**, клавиша M. После этого на экране будут отображаться лишь маркеры на чёрном фоне. Следить за поведением маркеров зачастую удобнее именно так.

Важный момент: маркер, дойдя до края видео, должен «погаснуть»! Однако иногда он не отключается, предпочитая вместо этого «зависнуть» на границе. То есть сам маркер уже почти не двигается, не следит ни за какой конкретной точкой (ибо точка давно «уплыла» за пределы кадра), но продолжает кормить Blender информацией (ясно, какой степени корректности). Этого нужно, очевидно, избегать. В режиме Mute footage увидеть такие проблемы существенно проще.

Есть способ быстро проверить качество трекинга. **OpenGL Render Animation.**



На таком рендере всё будет выглядеть в точности так же, как во вьюпорте:



Зато такой рендер идёт быстро и на нём более-менее видно, всё ли в порядке (не дёргается ли чего, не сползает ли...).

Советы, лайфхаки

Ниже — несколько советов по повышению уровня реалистичности и/или упрощения работы (из личного опыта).

Освещайте сцену с помощью HDRI-карты

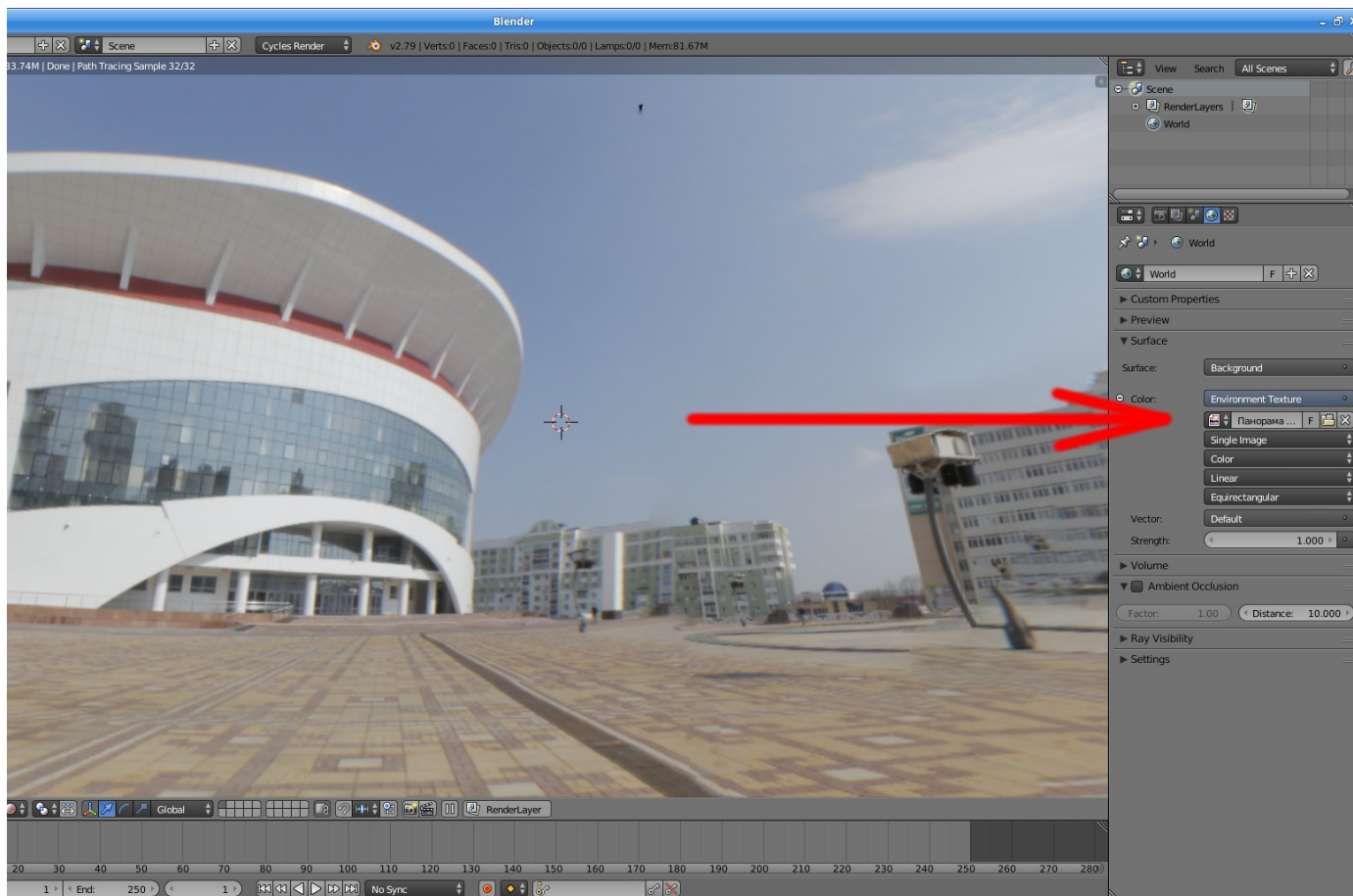


HDRI-карта — фотография с углом обзора в 360 градусов и в правильном формате (обычно —

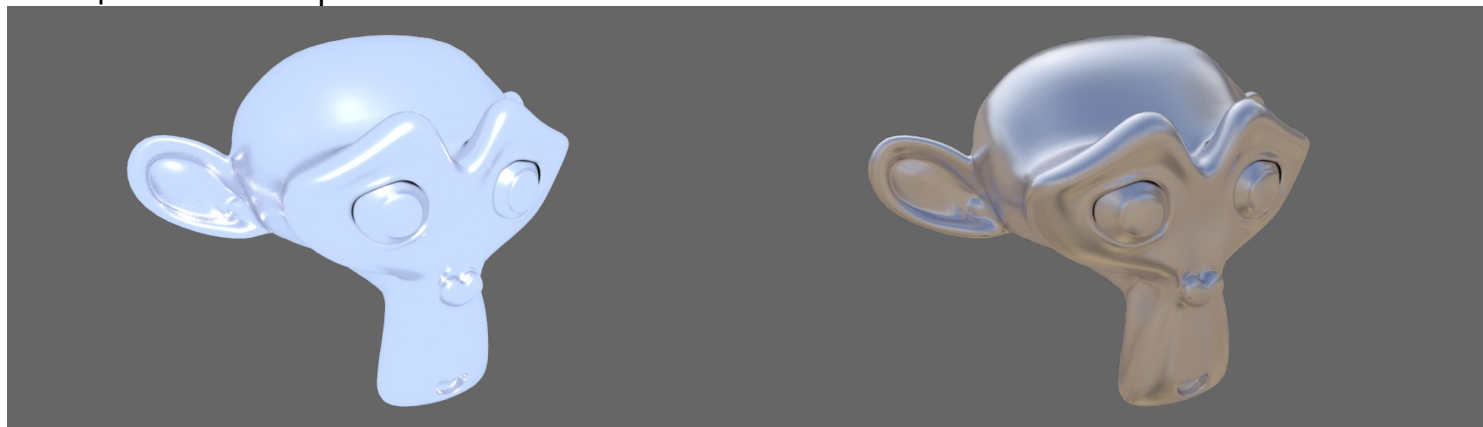
exr или hdr). В чём отличие от jpg?

Посмотрим на картинку выше. Здание справа — белое. Солнце — белое. С точки зрения jpg это здание и солнце — одинаковой яркости: и там, и там — белый цвет. HDR-форматы сохраняют информацию не только о цветах, но и о яркости, в частности, «понимают», что солнце гораздо ярче белой стены.

Панорама загружается в качестве Environment Texture:



Теперь мы имеем в сцене освещение, точно повторяющее освещение на месте съёмки. Для сравнения — модель, освещённая обычным белым окружением и лампой типа Sun и модель, освещённая HDRI-картой:



Разница налицо.

Как сделать HDRI-карту — тема отдельного урока, в интернете их при желании можно найти немало. Вкратце — понадобится сделать некоторое количество фотографий, которые охватят все 360 градусов вокруг. Сколько таких фотографий сделать — зависит от угла обзора объектива. 60-70 — если объектив обычный, 10-15 — если «рыбий глаз».

Также понадобится включить брекетинг (вилка) экспозиции, чтобы при каждом нажатии на

кнопку делалось 3 снимка — с обычной экспозицией, недоэкспонированный и переэкспонированный. Именно это позволит создать не просто сферическую панораму, а HDRI-карту.

Потом всё это надо сшить воедино. Из свободного софта могу посоветовать Hugin.

Не поленитесь допилить модели

Актуально для тех, кто использует выдранные из компьютерных игр модели.

Вышедший в 2004 году Half-Life 2 поражал всех своей реалистичной графикой, однако даже модели, вырванные из Half-Life 2, не очень-то реалистично смотрятся на фоне реального мира. А есть игры ещё более старые, есть стратегии с видом сверху (и соответствующим разрешением текстур)...

В общем, не поленитесь доработать вырванную из игры модель. Возможно, даже полностью воссоздать её с нуля. Это отнимет много времени и сил, но очень неплохо отразится на результате.

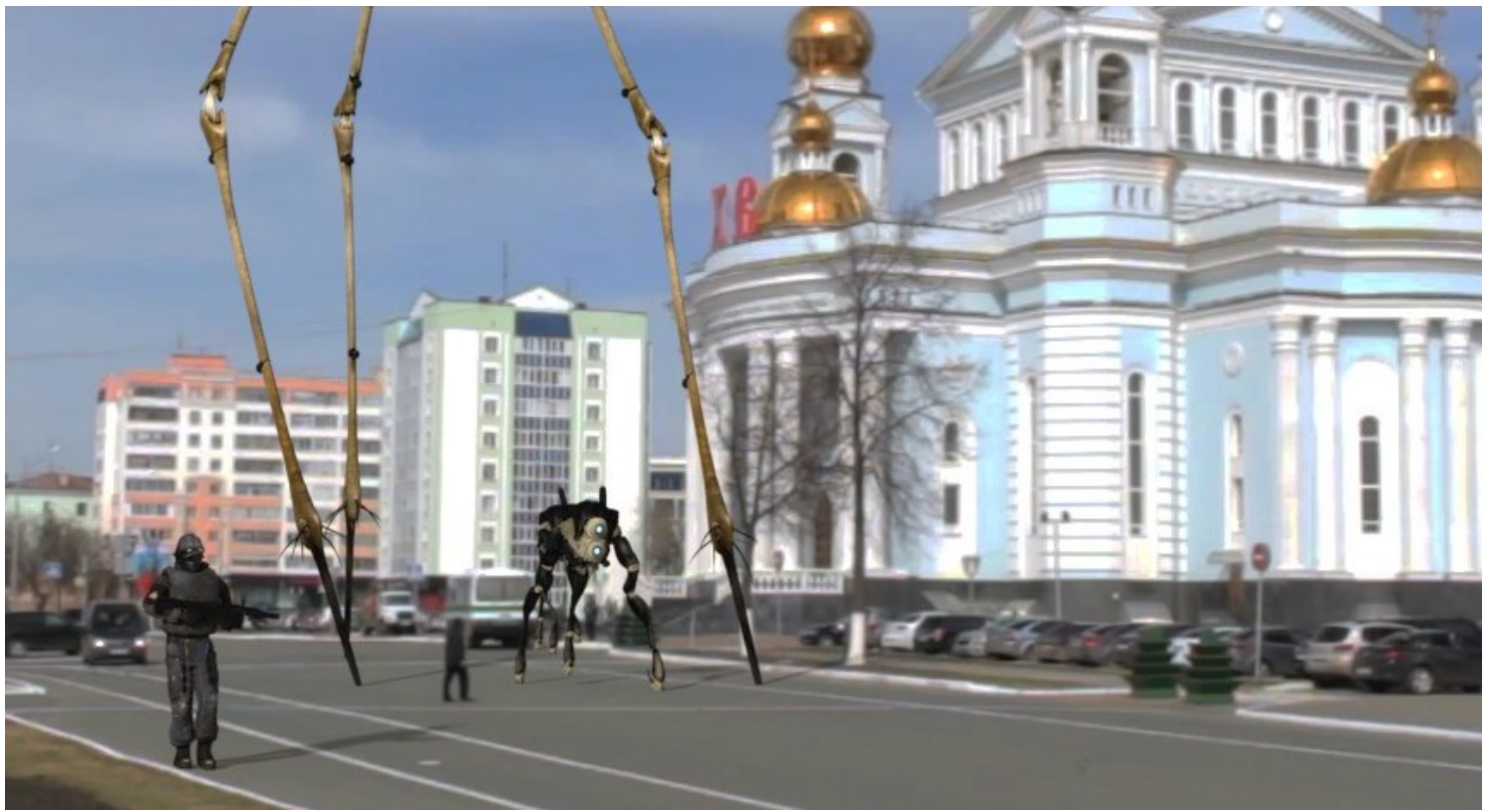
Для сравнения — БТР Альянса из Half-Life 2 и он же, переработанный в Blender.



Чёткость и размытие

Камеры несовершенны и мало какая из них способна сделать одинаково резким всё в кадре. Как следствие — объекты, на которые наведён фокус, более резкие, чем остальные.

Blender об этом не знает. Поэтому объекты, которые мы вставляем на видео, будут идеально резкими. Если в той области пространства, куда мы добавили объект, сей объект выделяется особой резкостью, в то время как всё вокруг (в этой области) имеет более размытые очертания — это бросается в глаза и выглядит неестественно.



Не слишком ли чёткие очертания у страйдера и охотников по сравнению, например, с расположенным рядом деревом или дорожной разметкой?

Чтобы исправить ситуацию, можно воспользоваться нодами Blur и Vector Blur (последний используется для размытия движением и добавляется автоматически при нажатии кнопки Setup Tracking Scene), а также разделом Depth of Field в настройках камеры в Blender.

Использование спикеров для озвучки

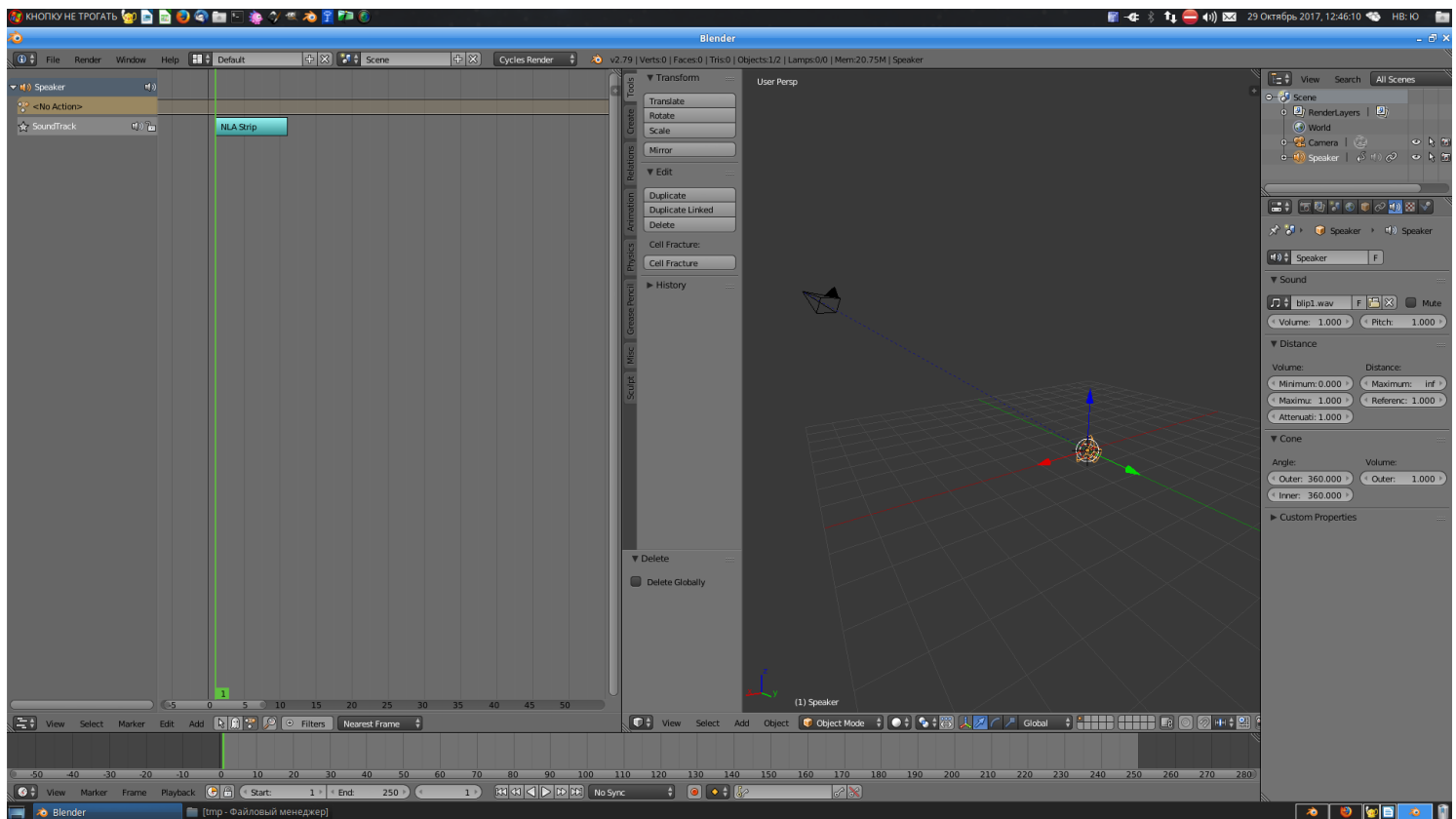
В Blender есть интересный объект — **Speaker**. Он позволяет создавать источник звука, причём этот источник будет находится в определённом месте и звучать по-разному в зависимости от расположения относительно камеры. Его можно использовать для озвучивания движущегося объекта, например (по мере удаления объекта от камеры звук должен также удаляться).

Порядок работы:

0. Parent'им спикер к нужному объекту;
1. выбираем спикер, потом, зажав Shift, выбираем камеру, жмём Ctrl+T, Track To Constraint. Теперь спикер прикреплён к объекту, издающему звуки, а также всегда направлен в сторону камеры.
2. В настройках спикера жмём Open и выбираем нужный нам звуковой файл.

Проблема в том, что звук начнёт проигрываться сразу, с первого же кадра. И проиграется только один раз. Как быть, если надо это изменить? Скажем, надо сделать, чтоб звук проигрывался трижды - на 30-ом кадре, на 72-ом и на 128-ом?

Сделать это просто. Открываем NLA Editor (а раньше вы часто им пользовались? Вообще знали, что такой есть?) и сразу же видим и наш спикер, и полоску, обозначающую звук (длина полоски соответствует длительности аудиофайла).



Просто выбираем эту бирюзовую полоску и дублируем (Shift+D)/двигаем (G) её так, как нам требуется.

