Multiscreen casting – 2nd semester

Administrative part at the start of the 2nd semester

- Your team will receive feedback from the TAs on the result of the 1st semester.
- You should discuss your results and the evaluation with your team. What can be done to correct standing bugs or implement missing functionality? What is still on your team's TODO or FIXME list?
- Your team should make a plan to fix or correct problems from the first semester and discuss this plan with the TA. This plan should be written down by the end of the first session.
- After this internal evaluation: reassign the jobs in the team (CEO, CTO, TC3, TC4, CR1, CR2). Everybody needs to switch to a new role. If your role in the previous semester was one of CEO or CTO then you need to pick one for this semester which has a digit in its abbreviation, otherwise your new role should not have the same two initial letters. You should communicate the new roles to your TA. See further down for the descriptions of TC3 and TC4.
- Consult the project description of the first semester if you need a reminder of what the different jobs in the team are supposed to be. Grading of these jobs will happen in the same way as for the first semester.

Three parts

The assignment of the 2nd semester consists of three parts:

- Part 1: Scientific tests: testing some of the assumptions made in the first semester by different teams.
- Part 2: Iterative development: adjust your implementation, fixing bugs, adding missing functionality, and possibly reimplementing things from scratch.
- Part 3: New features: developing and implementing some new features.

Note: the new features are only a part of this assignment as we think it is very important to reiterate the developments of the first semester! In "Part 1" we want every team to do some scientific tests on synchronization and color detection. We hope that some of these tests might reveal how to make improvements to your code from the first semester. Fixing bugs and implementing missing features happens in "Part 2". In particular, we want you to pay extra attention to monitoring and diagnostics of the algorithms. A lot of the time when a demo went wrong it was totally unclear to the team what could have happened. So in reiterating your code from the first semester we will ask you to implement more technical feedback on the master device explaining what is going on in the different steps of your algorithms. Lastly, in "Part 3" we propose some new and challenging features. Your team will have to be creative and investigate possible solutions. The final challenge will be an optional challenge where you can blow us away with your own ideas.

Part 1: Scientific tests

Code in this section should be written "from scratch" and is meant to be standalone from the multiscreen caster app. The code here is meant to test some standing assumptions. There are two topics on which we want to run some tests. Divide your team into two and assign each part to one of these topics. The topics are associated with one of the two TC's for this semester (see further) and the team member assigned to that specific TC should of course take part in the associated topic.

We call this part "scientific tests". This means you have to fully describe the circumstances and make sure that your results are repeatable and reproducible by an independent person.

We are fully aware that some teams have chosen different solutions in which the problems mentioned below are not present. But all teams have to make this part even if some teams already did similar tests. If you are in that case, consider it a head start.

Test 1: Browser synchronization (TC3)

A lot of teams explained in detail their algorithm for synchronizing the count down and animation tasks from the first semester with multiple slaves. We want to revisit some of the claims that we have heard.

NTP

Several teams have explained the Network Time Protocol (NTP) to us and how they have made a similar implementation to synchronize their browser apps. From the Wikipedia page on NTP we quote the following: "*NTP can usually maintain time to within tens of milliseconds over the public Internet, and can achieve better than one millisecond accuracy in local area networks under ideal conditions*". Sounds good, but, we want to remark that NTP is using UDP and not TCP to establish its protocol (and for good reason, please investigate if you do not immediately know what we mean...) while Socket.io, which those teams are using, is not UDP. Would this matter? We have seen some teams doing this successfully, so, maybe a better question is: how can this be done successfully? What is the accuracy that can be achieved?

Device clock

However, this also feels like reinventing the wheel. We think that almost all modern operating systems are already using some kind of NTP to synchronize the computer's clock to be as accurate as possible. Having an inaccurate clock is in fact a security risk in many situations. Also your smartphone is trying hard to keep accurate time and can synchronize its clock through multiple channels. When we asked teams how much they saw clocks being off we sometimes got some quite big numbers. So, part of this test will be to gather some statistics on how accurate the time of the clock on the different computers you have access to is (and noting what OS, what settings (is NTP being used?), what hardware, etc...), and similarly for smartphones and maybe tablets. Maybe some OSs / hardware platforms do a better job than others?

Let's do a test...

It is illustrative to open your browser and go to https://time.is which shows you the current time and an estimate of how accurate your computer clock is; accessed by javascript, from within a browser and displaying this in your browser through the javascript eventloop. This looks like a javascript NTP implementation. We could consider this big clock display as an animation which gets updated every second, which is not very often. But it could be used to check if multiple devices next to each other could do this accurately.

NB: Should we be worried about how https://time.is is trying to estimate offsets? If this runs in the browser, then they are using TCP and not UDP? Do we trust their result? You should at least convince yourself that they are trying to compare an atomic internet clock to the clock in your computer using the developer console of your browser such that their phrase "Your time is exact!" does not just mean that the time displayed in the browser is exact, but that actually your computer clock was found to be quite accurately exact. Find the javascript file which does the actual job. Place some breakpoints, play around...

A simple algorithm?

Let's for a minute assume that indeed all your slave devices have quite accurate time. A possible algorithm for the count down and animations could then be as follows: at time T0 the server sends a command to all slaves to execute an animation loop starting at time T1, taking into account the delay of delivering this message to the slaves (i.e., T1 – T0 >= delta). The slaves start their animation cycle at T1 (taking into account knowledge of the eventloop of the browser!). The only thing you have to guarantee now is that your animation runs smoothly and at the correct cue points. No communication with the server needed (except maybe for very long running animations). We suspect that a lot of the not so synchronous demo's that we saw were either caused by teams who did not have enough control on local animations (and this is a single browser problem and has nothing to do with multiple slave browsers) and/or who tried to adjust when to start the animation based on estimating delays between clients and server. As we pointed out in the assignment of the first semester, it is important to realise that javascript is executing your code synchronously. So when doing an animation it is important to know what the limitations are and how to get it to run smoothly.

NB: If indeed the computer clock is off by some amount and this can be measured accurately enough, then the above strategy still sounds like a good option. The time difference would only need to be measured once, e.g., when the page loads, and then the same algorithm can be used.

Tasks

1. **[Task ST1.1]** Investigate how accurate most computers and smartphones in your environment know the current time. This is a survey. Decide on what data to collect and how many samples (devices) you want. Decide on how to present the results. You will need to think about how to measure the difference with some absolute reference clock. You might only be able to accurately measure this if you are the owner of the device. You might need to have several columns per device for different measuring techniques (from very accurate to less accurate). If there are random errors then you might need to take multiple measurements and average them or take the median. How accurate do you need these results? How many measurements do you need per device? Make sure to describe this procedure.
   => Upload report of about 5 pages by 25/2 on Toledo. Put the names of the team members on the report of who is doing the "Test 1" part (about half of the team).
2. **[Task ST1.2]** Make a single browser animation for which you can choose the updating frequency as well as the workload it takes to generate the next frame (e.g., by a busy loop). How often do you need to update the animation to get a smooth motion? Investigate the best way of doing this, keeping in mind the browser eventloop (see references). We remark that not all animations are equal, e.g., updating text might trigger a reflow of all the text on the page by the browser (this could also happen for images), and loading a new image from a URL involves network communication. Make sure you understand the eventloop. Now decide on a procedure to verify how accurate the animation in the browser is timed and determine possible causes of disruption. Collect data on different browsers, operating systems and hardware. This is your base case. In a second phase you implement a way to have different browsers and/or different devices to start the animation sequence at the same time. Determine what the accuracy of such a setup is and under what assumptions this works. We expect you to perform the second test on a larger scale, e.g., ask a couple of neighbouring teams to participate in your final experiment using their phones and laptops.
   => Upload a report of about 5 pages by 10/3 on Toledo. Put the names of the team members on the report who are doing the "Test 1" part (about half of the team).

References

1. https://en.wikipedia.org/wiki/Network_Time_Protocol
2. https://time.is (open this in multiple browsers and multiple devices and see what happens)
3. https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/setTimeout and https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/setInterval and https://developer.mozilla.org/en-US/docs/Web/API/Window/requestAnimationFrame
   (especially read about accuracy and throttling and keeping your update code fast enough)
4. https://flaviocopes.com/requestanimationframe/

Test 2: Color recognition / Chroma keying (TC4)

A lot of teams had trouble detecting and identifying slave devices in the pictures from the master device. There are different reasons for these difficulties. One reason is our understanding of color and the attempts to use it to detect and identify slave screens. Another reason is noise and artefacts (and moving references for subsequent pictures). We will revisit both in this section, but we are not concerned with any of the further algorithms that were applied to solve the tasks of the first semester. We remark that it was, and is, not necessary to use color in detecting and identifying slave devices in the multiscreen caster app, so teams who implemented different methods do not need to change.

Thresholding pixel values

Our basic aim here is to be able to detect a large area of a particular color in a picture. Wouldn't it be nice if you told the slave device to fill the screen with rgb(0, 0, 255), you take a picture with the master device, and in your algorithm you just threshold your way through to find all those pixels? Unfortunately in practice the color will not everywhere be equal, nor will it have the same pixel values as we intended it to be. The first item in the reference list at the end of this section is a nice illustrative story and a partial solution, ending up with a formula of g * max(g-r, 0) * max(g-b, 0) to detect "green". (We would in fact be happy to read such a documented story to understand how different teams came to different solutions.) The wikipedia page on chroma keying lists some other (simple/naive) formulas of selecting a colour in rgb space (compare the formula (r + b) – g <= 0 to detect a green screen with the r, g, b lines in "Fig. 24" on the wikipedia page on HSL and HSV). Also the third reference in that list could be interesting and the most interesting solution on that page uses HSL to filter a color. We suspect a lot of the trouble in some of the solutions that we saw came from the inaccuracy of fixed thresholding. E.g., what happens if we turn off the lights? Or if we put the brightness of the screen very high? Or if we put a screen in "night mode" (which removes the blues in favour of "warmer" colors)? Or if the lights in the room have a very yellowish tone? Thresholds which work in one situation are not necessarily good for another situation.

Revisiting the original idea in the assignment of the first semester

The original assignment suggested using a very high contrast difference between two pictures (e.g., black and white) to detect the region of interest. In a simple implementation the two RGB pictures could be converted to grayscale, be normalized (this is scaling the values to use the full range from 0 to 255), and then pixels / regions could be compared between the two images w.r.t. their grayscale value. We think that you would intuitively agree that thresholding this difference (even with a fixed threshold) would work under the assumption that the slave devices have screens which are bright enough w.r.t. the environment. The key point here is that we are comparing relative things.

On the contrary, the idea of filtering on a specific shade of gray ("color" / value) in a single grayscale picture sounds like an impossible task. Sweeping through some ranges of thresholds would sweep through all pixels of the picture and we would have no reference as to where our intended value would end up in the values of the picture, leave alone all other pixels which might accidentally have the same value as we intended for our region of interest. If we aim the intended value of our region of interest to be 128 it could end up much lower if the brightness of the screen is very low w.r.t. the environment and also the opposite could happen. It could also be that the master device decides to run a non-linear filter to increase the quality of your picture...

Let's extend this reasoning to shades of blue. We ask the slave device to put its screen to be rgb(0, 0, 250). We then take a picture with the master device. Other environmental factors and the photo processing logic on the master device will change the color and the blue might turn out to be slightly purple. And even if we could ignore the other color channels (which is not what we should actually do) we have no control as to what the value of 250 would be mapped to in the final picture.

RGB vs HSL and HSV

One assumption that we could make is that all the automatic processing by the camera software on your phone has the aim of keeping the photograph look natural. So something which was blue in reality will hopefully still be blue in the picture. If there is a very bright blue somewhere else in the picture then we should assume that the relative difference between the very bright blue in the picture and our blue would be preserved. But even if we put the brightest blue on the screen we cannot guarantee that there is not a more bright area somewhere else in the picture. The biggest problem is that the rgb values for all three components will change dramatically while visually we just get a darker or brighter green. Try this for yourself http://hslpicker.com/; slide the hue picker to green and then change the saturation and lightness values while you observe what happens to the rgb values at the bottom (you can ignore the bottom slider which is the alpha, aka transparency, channel, and gets copied straight into the "a" value of rgba). So a partial solution to our problem could be to try and do thresholding on the hue component of HSL. Note that in this case the L

value should be not to the extreme left (black) or right (white), and the S value should not be too much to the left (gray). (Similar remarks hold for the S and V values of the HSV presentation.)

NB: We found several incorrect RGB to HSV or HSL javascript code on the web... Check before you copy paste and document your sources (and what you changed).

Dealing with noise and artefacts

When we consider noise in pixel values we should have an idea what types of disturbances we want to consider. Quite often one uses Gaussian blur filters with a certain radius to smear out disturbances over neighbouring pixels. This requires one to pick a radius. Such a filter will not solve the problem of flares, shadows or blocking. But we could make use of the things we control. We could for example demand that the area should take up at least 12% of the total image area. In combination with for example a picture size of 1280 x 720 this still leads us to an incredible amount of pixels that should make up that area. One way of dealing with this issue is to now divide the picture in little squares and determine the most likely color component by sampling some pixels in the square (either deterministically or randomly). Determining the most likely color can be adjusted for our purposes. E.g., if working in HSL we could ignore extremely dark or extremely light pixels; if we are looking for green we could count the number of sampled pixels fulfilling our "green" criterion (e.g., the one from wikipedia).

Tasks

1. **[Task ST2.1]** Investigate the accuracy (or inaccuracy) of putting a color on a computer display, taking a picture and analysing what comes out. In this part we do not want to detect the actual area. We are just going to measure the outcome and try to (dynamically?) threshold the color. It is hard to control the circumstances of such a test, but you should try to control the circumstances as much as possible. This might be easier at home than in the computer labs. The idea of these tests is to try and find out the worst possible cases. In particular we like you to test the following: 1) use one color (which color's work well, which work bad?) for the full screen and zoom into the full screen for the picture, 2) use one color for the full screen but allow more and more of the environment (describe the environment), 3) divide the screen in two and use two colors (what is the second color?) and repeat the previous tests. Try this in different environment settings (different rooms/lights) as well as different computer screens and also adjust the brightness of the screens. Try to come to a conclusion of what is a realistic way of detecting such a colored area. Think up front about how you want to present the data and your conclusions. Anticipate what you think will happen. Will you use RGB or HSL or HSV or something else?
   => Upload report of about 5 pages by 25/2 on Toledo. Put the names of the team members on the report of who is doing the "Test 1" part (about half of the team).
2. **[Task ST2.2]** In the second test we are going to include our knowledge of looking for a large enough area of a given color which might be affected by noise and other artefacts. Under the assumption that noise and small artefacts are random we will determine if a certain area in the picture is part of the color by sampling some pixels in this area and combining their outcome in a single value. This value could be simply 0 or 1; or, in a different implementation any value between 0 and 1. Again, we will repeat similar tests under controlled circumstances. One way of presenting single test results is to subdivide the picture into little squares and denote the value for each square by a transparent mask.
   => Upload a report of about 5 pages by 10/3 on Toledo. Put the names of the team members on the report who are doing the "Test 1" part (about half of the team).

References

1. https://blogs.mathworks.com/steve/2014/08/12/it-aint-easy-seeing-green-unless-you-have-matlab/
2. https://en.wikipedia.org/wiki/Chroma_key#Programming
3. https://stackoverflow.com/questions/4063965/how-can-i-convert-an-rgb-image-to-grayscale-but-keep-one-color/
4. https://en.wikipedia.org/wiki/HSL_and_HSV

5. https://stackoverflow.com/questions/38419980/how-to-accurately-filter-rgb-value-for-chroma-key-effect
6. http://hslpicker.com/

Part 2: Iterative development

A large part of the work for the second semester will be on iterating over the code which was used for the demos of the first semester. Fixing bugs and implementing missing features should have the highest priority. Particular attention will go to implement more technical feedback on the master device to illustrate what is going on in the different steps of your algorithms. The idea is that this should make it easier to detect bugs or understand external influences.

A change of season

We have switched roles inside the team, but it might also be a good idea to have team members iterate over code which they did not develop themselves (they might however have done a code review on it). They will try to understand the code and try to trace the bugs.

The state of the code

We will ask you to look at the code reviews from the previous semester, take into account the feedback from the TAs on the final demo from last semester, gather your TODOs and FIXMEs, and discuss open problems.

Tasks

1. **[Task ID1]** A plan of attack should be conceived and discussed with the TA. This plan should list explicitly what you are trying to achieve / fix, how, by whom, and a timeframe.
   => Upload your plan of attack (maximum 2 pages) by 18/2 on Toledo. Also include how the new jobs in the team have been assigned.
2. **[Task ID2]** Demo in the middle of the semester which shows an upgraded version of the demo at the end of the first semester.
   => Demonstration on 24/3 or 31/3. The demonstration will be similar to the two demonstrations of the first semester, but with particular attention to the master device clearly showing all the steps of the detection and identification algorithm.

Part 3: New features

We will extend the framework from last semester with some extra features. We leave you with lots of choices of how to do things in this section. Note that new features build upon a working app from the first semester, so you need to consider parts 1 and 2 first!

Feature 1: Moving master: A change of perspective

In this feature the slaves will show their part of a 3D scene from the perspective of the master device. If the master device moves then the views on the slave devices get updated. In its simplest form this trick could also be done with a single screen.

Extension of the first semester

This feature has to be considered as an extension to the app you developed in the first semester. (Of course you should reiterate its design and implementation in part 2.) Building directly on the functionality of the first semester you could change the perspective of the master device on the triangulation, the animation and the picture. Note that the perspective on the picture also works in case of a single slave device.

Tracking perspective: sensors and visual clues

To be able to successfully track the perspective you will have to develop a new algorithm. There are many ways to tackle this problem. For example, you could try to use the sensors in the smartphone which acts as the master device. A good point to start with is figuring out what information is available (in HTML5 inside your

browser) and how accurate it is. For example, a smartphone has an accelerometer, but how well can you compute changes in position using the accelerometer? If some experimentation is done, make sure it is reproducible and report this in a scientific way.

Another piece of information that is available might be the slave screens. Could these help to track position changes of the master? Think of possible algorithms and decide if and how you want to implement this. Again here as well, we are interested in the accuracy you can achieve and we expect reproducible experiments reported in a scientific way.

Yet another option is to use an algorithm to calculate "cue points" in the images / video captured. This means you would need a fast algorithm that can calculate cue points and quickly determine the changes to the position of your master device. Again, how accurate can this be done? Experiments should be reproducible and properly reported.

Combining different estimators and smoothing motion

Finally we want to mention that it is of course possible to combine different methods and take into account their accuracy. If you opt for such an option then we want to see a mathematical formulation of how you combine your different estimates to make a more accurate estimate.

In the case of single or multiple estimators you might find it necessary to smooth out the motion and avoid small oscillatory movements of your perspective views.

Panic buttons

Out of previous experiences, we know that the results might sometimes be quite inaccurate. We allow you to put controls on the master device to reset the position, or stop an incorrect continuous motion if the device is standing still (e.g., you could implement a "reset" position button, or a "no motion" button).

Visualising a 3D scene

The ultimate goal here is to render a 3D scene on the slave devices and use the master device to change perspective. A 3D scene can be anything of your choice. The more realistic it is, the better, e.g., a forest, a city, a field with different plants. But think about the computational cost to generate this scene and how efficient it can be recomputed when the perspective changes. A very popular way to generate plants, for example, are L-systems (fractals). These require only a small amount of memory, since they are computed on the go.

An important property of effective 3D renderings is shading. So if you want to give your scene a realistic feeling, then this is one of the first things you could incorporate.

Tasks

1. **[Task F1.1]** Add perspective tracking for the master device. This should be a proof of concept. Try to make a minimalistic setup in which you experiment and show that it works. Which algorithms you implement to do this is up to your team. A lot of different and fancy things can be done here, but we leave it up to your team how to decide where the effort needs to go. Keep in mind that a working version of the first semester results is required for this task to function, so your main effort should probably be in Part 2!
   => Upload a report of about 5 pages by 28/4 on Toledo. A demo will be requested in the labs.
2. **[Task F1.2 / optional]** If your team is running smoothly, then this task adds rendering a real 3D scene. It is up to your team to decide the particular design and implementation aspects. Completing this task successfully will rank you in our top team list.

References

We are not giving you any references here. Investigate thoroughly and put a sufficient number of references in your reports. Good references make it easy to convince us that your approach makes sense...

Feature 2: Blow us away by this bonus feature! (optional)

In case your team is reading this assignment and by some magic discovered that you already implemented all of this in the past weekend, that all bugs have been fixed and everything is running seamlessly, ... then we challenge you to blow us away with your own ideas. Show us a game or show us a live version of this commercial: https://www.youtube.com/watch?v=7Rr9iXg9wsM (which we used in the opening slides of the first semester). It's up to you. If you blow us away, you will be forever in the hall of fame.

Technical Committees for the 2nd semester

The technical committees have the same function as in the first semester and we will start TC meetings from the second week of the semester. Since both TCs are related to the two different topics in the "Scientific tests" parts they could decide to jointly discuss what kind of procedure to follow to gather data and what data to gather. The working in the technical committees is the same as for the first semester. All TC members will have to give a technical presentation in their TC on some technically relevant topic and will be graded.

TC3: Synchronization and communication

This TC will discuss topics related to synchronization and communication between slave devices, the master device and the server.

TC4: Detection and identification

This TC will discuss topics related to identifying slave devices in a picture. Additionally it can also discuss other image related algorithms, e.g., for detecting "clue points".

End of semester demo, presentation and report

Like last semester there will be a final report at the end of the semester, and a presentation with demo in the final two weeks of the semester.

=> Upload the final report by 12/5 on Toledo.

=> Presentations and demo are planned in the last week of the semester (week of 18/5).

Correctly referring to other people's work and "borrowing" images or code

You cannot just "copy-paste" things that you find on the internet or get from friends. Whenever you modify code that you found online, e.g., on Stack Exchange, you are required to put the URL in your code. Whenever you use an image (e.g. in your user interface, or to illustrate your report or your slides) you have to check if the usage is allowed and correctly cite the source. You are required to check software licenses of all the software that you use and make a list. Think about what those licenses imply for your software project and put a license on your project as well.

Especially in your report we expect a list of references. For websites this includes the URL and the date that you visited the website. In your slides we expect to see a correct citation whenever you use an illustration. In your application we expect a credits screen which shows your license and credits for whatever software you are using, or images you are using.