

Task 1 | GIMP | “Filters”

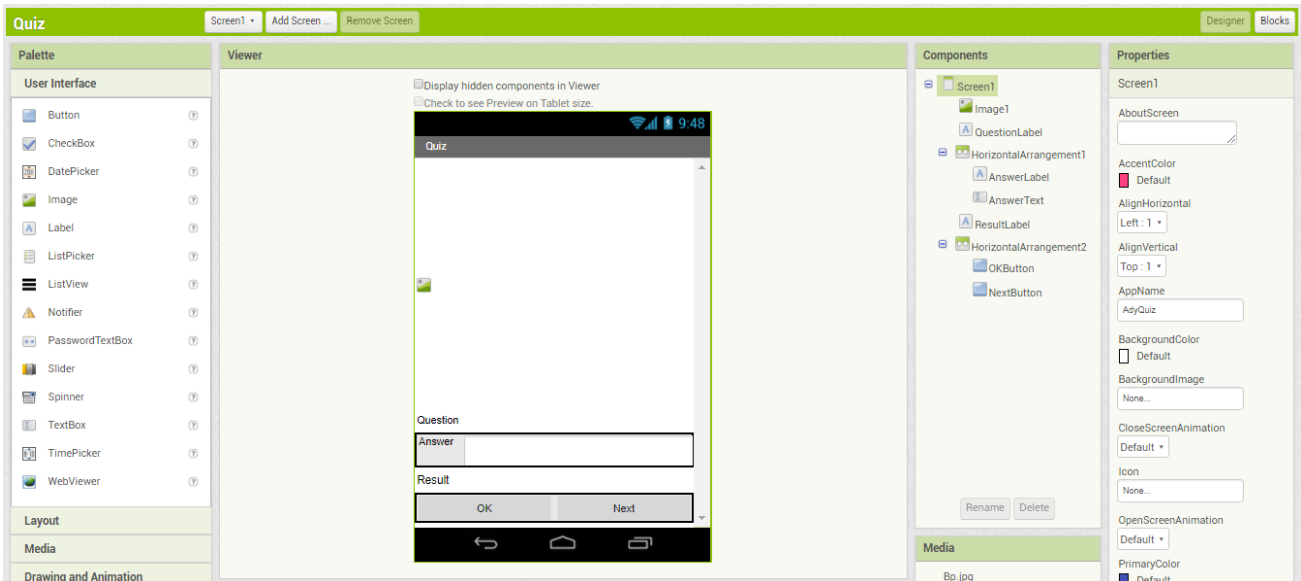
- By using filters, we can set decorative effects to our image to make it looks better.
 - If the image consists of one layer only, and there is no selected area, the filter will modify the whole image.
 - If the image consists of more layers, and there is no selected area, the filter will modify the currently active layer only.
 - If there is an area selected on the image, the filter will modify the selected area only.
- Choose an image on the following website: <https://apod.nasa.gov/apod/archivepix.html>
- Download the chosen image to your computer and create your own, totally unique image by adding one or more filters to it. Click on the *Filters* option and have fun. 😊
- You can find some examples for using filters below.



- Save your project as *Filter01*, *Filter02* etc. in *.xcf* and also export the image in *.png* format.
- Find out more about *Filters* by visiting: <https://docs.gimp.org/en/filters.html>

Task 2 | MIT App Inventor 2 | "Quiz 1"

- Here are the components for the *Quiz 1* app, as shown in the *Component Designer*:



- Create 3 variables consisting of 3 lists.
 - In the first one, have to be stored the *Questions*.


```

                    initialize global QuestionList to
                    make a list
                    "What is the capital city of Hungary?"
                    "What is the climate of Hungary?"
                    "What was the lowest temperature"
                    "What was the highest temperature"
                    "How many grams does the smallest native bird to ..."
                    "How many meters is the perimeter of the thickest..."
                    "How tall is the lowest point of Hungary?"
                    
```
 - In the second one, have to be stored the *Correct answers*.


```

                    initialize global AnswerList to
                    make a list
                    "Budapest"
                    "continental"
                    "-35"
                    "42"
                    "5"
                    "12"
                    "76"
                    
```
 - In the third one, have to be stored the *Images* for the questions.


```

                    initialize global PictureList to
                    make a list
                    "Bp.jpg"
                    "seasons.jpg"
                    "coldest.jpg"
                    "warmest.jpg"
                    "smallest.jpg"
                    "tree.JPG"
                    "lowest.jpg"
                    
```
- Create 2 more variables.
 - One of them has to store the number of the current question.


```

                    initialize global currentQuestionIndex to 1
                    
```
 - The other one has to store the number of the correct answers.


```

                    initialize global correct to 0
                    
```

- At the start of the application the *QuestionLabel* has to display the first element of the *QuestionList* and also the belonging image.

```

when Screen1.Initialize
do
  set QuestionLabel.Text to select list item list get global QuestionList index 1
  set Image1.Picture to select list item list get global PictureList index 1
    
```

- When the *OKButton* is clicked, it has to analyze if the entered text is in accordance with the current element of the *AnswerList*.
 - If yes, the *ResultLabel* has to show the correct answer and increase the number of the correct answers by adding plus one point.
 - Else, it has to show the “Think about it!” text.

```

when OKButton.Click
do
  if AnswerText.Text = select list item list get global AnswerList index get global currentQuestionIndex
  then
    set ResultLabel.Text to "Correct"
    set global correct to get global correct + 1
  else
    set ResultLabel.Text to "Think about it!"
    
```

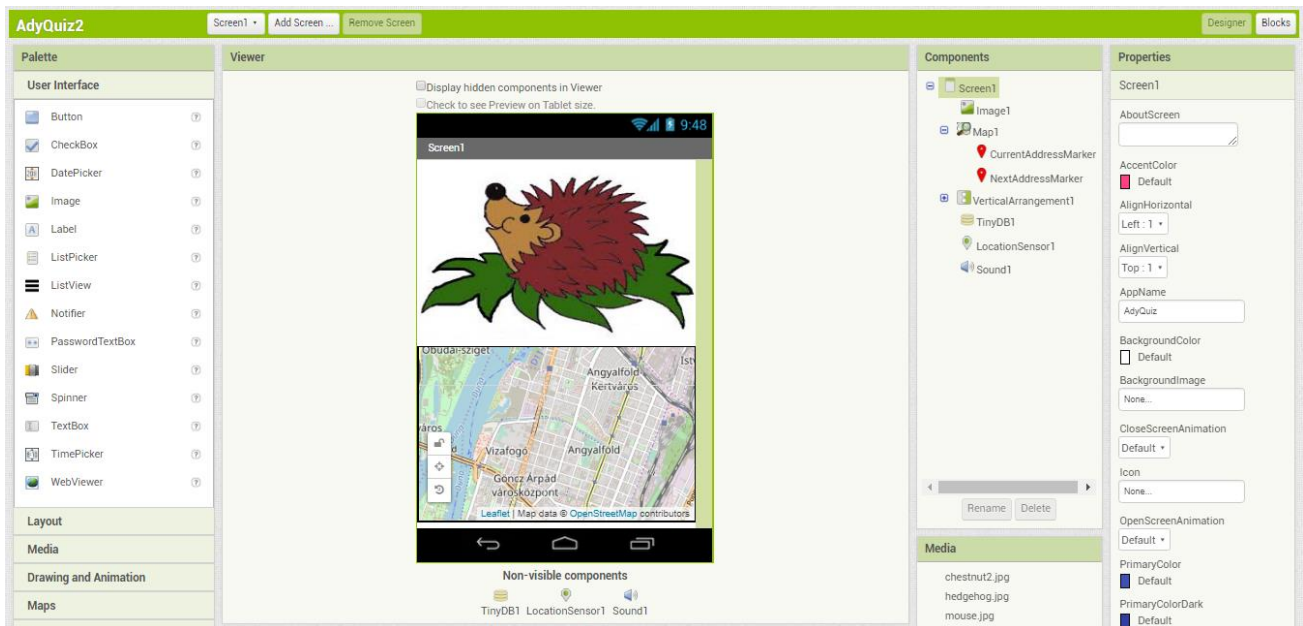
- When the *NextButton* is clicked, it has to analyze if the number of the question is smaller than the index of the last element of the *QuestionList*.
 - If yes, it has to increase the number of the questions by adding one more, so the *QuestionLabel* has to show the next element from the *QuestionList* and also the belonging picture.
 - Else, it has to turn off the *AnswerTextBox* by setting the *HorizontalArrangement1* and *HorizontalArrangement2* (which contain the buttons) to invisible.
 - It has to show the “BYE” image and the number of the correct answers.
 - And it has to set the *AnswerText* and the *QuestionLabel* to empty.

```

when NextButton.Click
do
  if get global currentQuestionIndex < 7
  then
    set global currentQuestionIndex to get global currentQuestionIndex + 1
    set QuestionLabel.Text to select list item list get global QuestionList index get global currentQuestionIndex
    set Image1.Picture to select list item list get global PictureList index get global currentQuestionIndex
    set ResultLabel.Text to ""
  else
    set HorizontalArrangement1.Visible to false
    set HorizontalArrangement2.Visible to false
    set Image1.Picture to "bye.jpg"
    set ResultLabel.Text to join "You have answered " get global correct " question correctly."
  end if
  set AnswerText.Text to ""
  set QuestionLabel.Text to ""
    
```

Task 3 | MIT App Inventor 2 | “Ady Quiz”

- Here are the components for the *Ady Quiz* app, as shown in the *Component Designer*:



- Declare the following variables:
 - currentQuestionIndex*: shows where we currently are.


```
initialize global currentQuestionIndex to 0
```
 - currentQuestionText*: elements of the database (by tags) which are set in list.


```
initialize global currentQuestionText to make a list ["Point1", "Point2", "Point3", "Point4", "Point5"]
```
 - questionData*: the data of the current point will be stored here.


```
initialize global questionData to create empty list
```
 - lat* and *lon*: the coordinates of the destination.


```
initialize global lat to 0 initialize global lon to 0
```
 - latCurrent* and *lonCurrent*: the coordinates of the player's current positions.


```
initialize global latCurrent to 0 initialize global lonCurrent to 0
```
- Turn the *LocationSensor* on and define the current position of the player.

```
when LocationSensor1.LocationChanged
  latitude longitude altitude speed
do
  set LocationSensor1.Enabled to true
  set global latCurrent to get latitude
  set global lonCurrent to get longitude
  call CheckLocation
  set LocationSensor1.Enabled to true
```

- Check the position of the player to know how far the player is from the destination.
 - If the player is close to the right place the phone has to be fluttering, the map has to disappear and the picture, belonging to that point, has to show up.
 - Then the *QuestionLabel* and the multiple choice answer buttons have to show up.
 - The *ResultLabel* has to be set blank.

- If the player isn't close to the point, the map has to show the current position of the player.

```

to CheckLocation
do
  if
    absolute
    get global lon - get global lonCurrent < 0.0002 and
    absolute
    get global lat - get global latCurrent < 0.0002
  then
    call Sound1 .Vibrate
    millisecs 2000
    set Map1 .Visible to false
    set Image1 .Visible to true
    set VerticalArrangement1 .Visible to true
    set HorizontalArrangement1 .Visible to true
    set QuestionLabel .Text to select list item list get global questionData
    index 1
    set AnswerAButton .Text to select list item list get global questionData
    index 2
    set AnswerBButton .Text to select list item list get global questionData
    index 3
    set AnswerCButton .Text to select list item list get global questionData
    index 4
    set Image1 .Picture to select list item list get global questionData
    index 6
    set ResultLabel .Text to " "
  else
    call CurrentAddressMarker .SetLocation
    latitude get global latCurrent
    longitude get global lonCurrent
    
```

- At the start of the application, the value of *currentQuestion* has to be set to 1.
 - Store the items of the *Points* with the *StoreValue* procedure.
 - The first one has to be the defining tag, then create a list consisting of the belonging question, the optional answers, the index of the correct answer, the image, and the GPS coordinates.
 - Store the items to each *Points*.
 - Set the value of the *questionData* variable to the data of the current point.
 - Set also the coordinates of the destination.
 - Set the image and the buttons to invisible and set the map to visible. Set the center of the map to be the destination exactly.
 - Call the *CheckLocation* procedure.
 - Create markers on the map to show the destination and the current position.

```

when Screen1 .Initialize
do
  set global currentQuestionIndex to 1
  call TinyDB1 .StoreValue
  tag Point1
  valueToStore make a list
  It's just like a brush, but you can not comb wit...
  Hedgehog
  Anteater
  prickly devil
  2
  hedgehog.jpg
  47.55468
  19.091918
  call TinyDB1 .StoreValue
  tag Point2
  valueToStore make a list
  He drills, sculpts, knocks like a hammer. He is ...
  pigeon
  woodpecker
  parrot
  3
  woodpecker.png
  47.55371
  19.092358
  
```

```

call TinyDB1 .StoreValue
tag Point3
valueToStore make a list
Ball like a hedgehog, green in its color, autumn...
walnut
peanut
chestnut
4
chestnut2.jpg
47.552833
19.095244
call TinyDB1 .StoreValue
tag Point4
valueToStore make a list
His coat is gray with a long, thin tail, his del...
cat
dog
mouse
4
mouse.jpg
47.555198
19.095475
  
```

```

call TinyDB1 StoreValue
tag Points
valueToStore make a list
    I wrap a net night and day, and I come up and do...
    spider
    ladybird
    ant
    2
    spider.jpg
    47.555386
    19.091972

set global questionData to call TinyDB1 GetValue
tag select list item list get global currentQuestiontext
index 1
valueIfTagNotThere

set global lat to select list item list get global questionData
index 7
set global lon to select list item list get global questionData
index 8

set VerticalArrangement1 Visible to false
set Image1 Visible to false
set Map1 CenterFromString to join get global lat
    *
    get global lon

call CheckLocation
call NextAddressMarker SetLocation
latitude get global lat
longitude get global lon

call CurrentAddressMarker SetLocation
latitude get global latCurrent
longitude get global lonCurrent
    
```

- If the *ShowNextButton* is clicked, the current position of the player has to be checked.
 - If the player hasn't approached the last point the visibility of the image, the question and answers have to be set to *false*.
 - Set the map to visible and the center of it to be the current point exactly. Create a marker on the map to show the current point.
 - Then, switch to the next question. The data of it has to be retrieved from the database and the value of the *questionData* variable has to be rewritten.
 - The text of the *ResultLabel* has to be set to blank.
 - Retrieve and store the coordinates of the next destination and a marker has to show its exact location on the map.
 - If the player has approached the last point, the visibility of the image, the question and the answers have to be set to *false*.
 - Then, set the *EndButton* to visible and show the image with the name '*the-end-png.png*'.

```

when ShowNextButton Click
do
if get global currentQuestionIndex < 5
then
set Image1 Visible to false
set VerticalArrangement1 Visible to false
set HorizontalArrangement1 Visible to false
set Map1 Visible to true
set Map1 CenterFromString to join select list item list get global questionData
    index 7
    *
    select list item list get global questionData
    index 8

call CurrentAddressMarker SetLocation
latitude get global latCurrent
longitude get global lonCurrent

set global currentQuestionIndex to get global currentQuestionIndex + 1

set global questionData to call TinyDB1 GetValue
tag select list item list get global currentQuestiontext
index get global currentQuestionIndex
valueIfTagNotThere

set ResultLabel Text to 
set global lat to select list item list get global questionData
index 7
set global lon to select list item list get global questionData
index 8

call NextAddressMarker SetLocation
latitude get global lat
longitude get global lon
    
```

```

else
  set VerticalArrangement1 . Visible to false
  set ResultLabel . Visible to false
  set EndButton . Visible to true
  set ShowNextButton . Visible to false
  set Image1 . Picture to the-end-png.png
  set Map1 . Visible to false
  
```

- Check the answer if it is correct or not with the same method at the 3 buttons.
 - If the button clicked is in accordance with the correct answer stored, the *ResultLabel* has to show *Correct!*.
 - Otherwise, it has to show *Check it up!*.

```

when AnswerAButton . Click
do
  if AnswerAButton . Text = select list item list get global questionData
  index select list item list get global questionData
  index 5
  then set ResultLabel . Text to Correct
  else set ResultLabel . Text to Look at it
  
```

```

when AnswerBButton . Click
do
  if AnswerBButton . Text = select list item list get global questionData
  index select list item list get global questionData
  index 5
  then set ResultLabel . Text to Correct
  else set ResultLabel . Text to Look at it
  
```

```

when AnswerCButton . Click
do
  if AnswerCButton . Text = select list item list get global questionData
  index select list item list get global questionData
  index 5
  then set ResultLabel . Text to Correct
  else set ResultLabel . Text to Look at it
  
```