Importing Datasets and Finding a Specific Game to Analyse

In [8]:
```
from mplsoccer import Sbopen
import pandas as pd

# instantiate a parser object
parser = Sbopen()
```

In [9]:
```
df_competition = parser.competition()
df_competition.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 75 entries, 0 to 74
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   competition_id             75 non-null     int64 
 1   season_id                  75 non-null     int64 
 2   country_name               75 non-null     object
 3   competition_name           75 non-null     object
 4   competition_gender         75 non-null     object
 5   competition_youth          75 non-null     bool  
 6   competition_international  75 non-null     bool  
 7   season_name                75 non-null     object
 8   match_updated              75 non-null     object
 9   match_updated_360          57 non-null     object
 10  match_available_360        11 non-null     object
 11  match_available            75 non-null     object
dtypes: bool(2), int64(2), object(8)
memory usage: 6.1+ KB
```

Find the competition ID for the Euros, found in the GitHub StatsBomb repositories

Euros Comp ID == 55

In [10]:
```
# Filter for Premier League only
euros_df = df_competition[df_competition['competition_id'] == 55]
print(euros_df)
```

```
    competition_id  season_id country_name competition_name  \
68              55        282       Europe        UEFA Euro   
69              55         43       Europe        UEFA Euro   

   competition_gender  competition_youth  competition_international  \
68               male              False                      True   
69               male              False                      True   

   season_name               match_updated           match_updated_360  \
68        2024  2024-09-28T16:51:20.698794  2025-03-24T14:12:30.785094   
69        2020  2024-07-31T12:29:15.702309  2024-07-31T12:30:57.587087   

           match_available_360             match_available  
68  2025-03-24T14:12:30.785094  2024-09-28T16:51:20.698794  
69  2024-07-31T12:30:57.587087  2024-07-31T12:29:15.702309  
```

In [11]:
```
print(euros_df.columns)
```

```
Index(['competition_id', 'season_id', 'country_name', 'competition_name',
       'competition_gender', 'competition_youth', 'competition_internation
al',
       'season_name', 'match_updated', 'match_updated_360',
       'match_available_360', 'match_available'],
      dtype='object')
```

Select the specific 2024 Euros competition

In [12]:
```
comp55 = df_competition[df_competition['competition_id'] == 55]
print(comp55)
```

```
    competition_id  season_id country_name competition_name  \
68              55        282       Europe        UEFA Euro   
69              55         43       Europe        UEFA Euro   

   competition_gender  competition_youth  competition_international  \
68               male              False                      True   
69               male              False                      True   

   season_name               match_updated           match_updated_360  \
68        2024  2024-09-28T16:51:20.698794  2025-03-24T14:12:30.785094   
69        2020  2024-07-31T12:29:15.702309  2024-07-31T12:30:57.587087   

           match_available_360             match_available  
68  2025-03-24T14:12:30.785094  2024-09-28T16:51:20.698794  
69  2024-07-31T12:30:57.587087  2024-07-31T12:29:15.702309  
```

Season ID == 282

In [14]:
```
matches = parser.match(competition_id=55, season_id=282)
```

In [15]:
```
print(matches['competition_stage_name'].unique())
```

```
['Semi-finals' 'Final' 'Quarter-finals' 'Group Stage' 'Round of 16']
```

In [16]:
```
final_match = matches[matches['competition_stage_name'] == 'Final']
```

Look for the match stats for only the Final of the 2024 Euros

Get the match ID for the Euro 2024 final

In [17]:
```
print(final_match[['home_team_country_name',
                   'away_team_country_name','match_id']])
```

```
  home_team_country_name away_team_country_name  match_id
1                  Spain                England   3943043
```

Collect useful info about the Euros final match 2024

In [19]:
```
Euro_Final = matches[matches['match_id'] == 3943043]
print(Euro_Final[['home_score', 'away_score', 'competition_name',
                  'competition_stage_name', 'home_team_name',
                  'away_team_name',
                 ]])
```

```
   home_score  away_score competition_name competition_stage_name  \
1           2           1        UEFA Euro                  Final   

  home_team_name away_team_name  
1          Spain        England  
```

Understand the Lineup dataset

In [20]:
```
df_lineup = parser.lineup(3943043)
```

In [21]:
```
df_lineup.shape
```

Out[21]:
```
(50, 9)
```

In [22]:
```
print(df_lineup.columns)
```

```
Index(['player_id', 'player_name', 'player_nickname', 'jersey_number',
       'match_id', 'team_id', 'team_name', 'country_id', 'country_name'],
      dtype='object')
```

Understand the Events dataset

In [23]:
```
df_events = parser.event(3943043)[0]
```

In [24]:
```
df_events.columns
```

Out[24]:
```
Index(['id', 'index', 'period', 'timestamp', 'minute', 'second', 'posses
sion',
       'duration', 'match_id', 'type_id', 'type_name', 'possession_team_
id',
       'possession_team_name', 'play_pattern_id', 'play_pattern_name',
       'team_id', 'team_name', 'tactics_formation', 'player_id', 'player
_name',
       'position_id', 'position_name', 'pass_recipient_id',
       'pass_recipient_name', 'pass_length', 'pass_angle', 'pass_height_
id',
       'pass_height_name', 'end_x', 'end_y', 'body_part_id', 'body_part_
name',
       'sub_type_id', 'sub_type_name', 'x', 'y', 'outcome_id', 'outcome_
name',
       'under_pressure', 'out', 'counterpress', 'pass_switch',
       'dribble_nutmeg', 'aerial_won', 'pass_cross', 'technique_id',
       'technique_name', 'pass_no_touch', 'foul_won_defensive', 'off_cam
era',
       'pass_assisted_shot_id', 'pass_shot_assist', 'shot_one_on_one',
       'shot_statsbomb_xg', 'shot_key_pass_id', 'goalkeeper_position_i
d',
       'goalkeeper_position_name', 'end_z', 'shot_first_time',
       'dribble_overrun', 'foul_committed_advantage', 'foul_won_advantag
e',
       'ball_recovery_recovery_failure', 'foul_committed_card_id',
       'foul_committed_card_name', 'block_offensive', 'shot_deflected',
       'block_deflection', 'foul_committed_offensive',
       'injury_stoppage_in_chain', 'substitution_replacement_id',
       'substitution_replacement_name', 'pass_goal_assist', 'pass_cut_ba
ck',
       'block_save_block'],
      dtype='object')
```

In [25]:
```
print(df_events[['x', 'y']].head(5))
```

```
      x     y
0   NaN   NaN
1   NaN   NaN
2   NaN   NaN
3   NaN   NaN
4  60.0  40.0
```

In [26]:
```
print(df_events['player_name'].unique())
```

```
[nan 'Kobbie Mainoo' 'Jordan Pickford' 'Bukayo Saka' 'Unai Simón Mendibil'
 'Robin Aime Robert Le Normand' 'Daniel Carvajal Ramos' 'Jude Bellingham'
 'Declan Rice' 'Daniel Olmo Carvajal' 'Álvaro Borja Morata Martín'
 'Lamine Yamal Nasraoui Ebana' 'Marc Guehi' 'Phil Foden'
 'Rodrigo Hernández Cascante' 'Aymeric Laporte' 'Kyle Walker' 'Luke Shaw'
 'Harry Kane' 'John Stones' 'Nicholas Williams Arthuer'
 'Marc Cucurella Saseta' 'Fabián Ruiz Peña' 'Martín Zubimendi Ibáñez'
 'Ollie Watkins' 'Mikel Oyarzabal Ugarte' 'Cole Palmer'
 'José Ignacio Fernández Iglesias' 'Ivan Toney' 'Mikel Merino Zazón']
```

In [27]:
```
df_events.shape
```

Out[27]:
```
(3312, 75)
```

With the events database found with the correct ID, various figures can be made to analyse the match visually and compare noth teams

Import necessary libraries for creating figures

In [43]:
```
from mplsoccer import Pitch
import matplotlib.pyplot as plt
import seaborn as sns
```

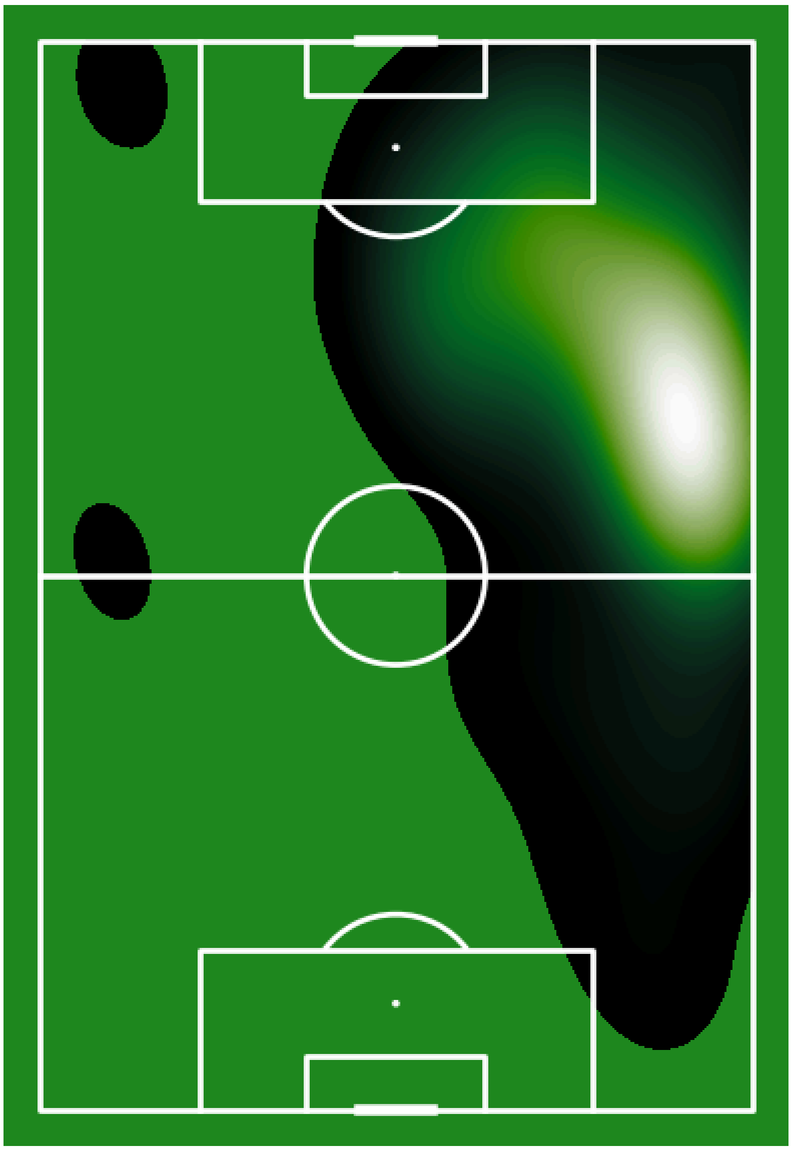
In [44]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from matplotlib.colors import LinearSegmentedColormap
import cmasher as cmr
from matplotlib.colors import to_rgba
from mplsoccer import VerticalPitch, Sbopen
from mplsoccer.utils import FontManager
```

Touch Heatmap of Specific Players

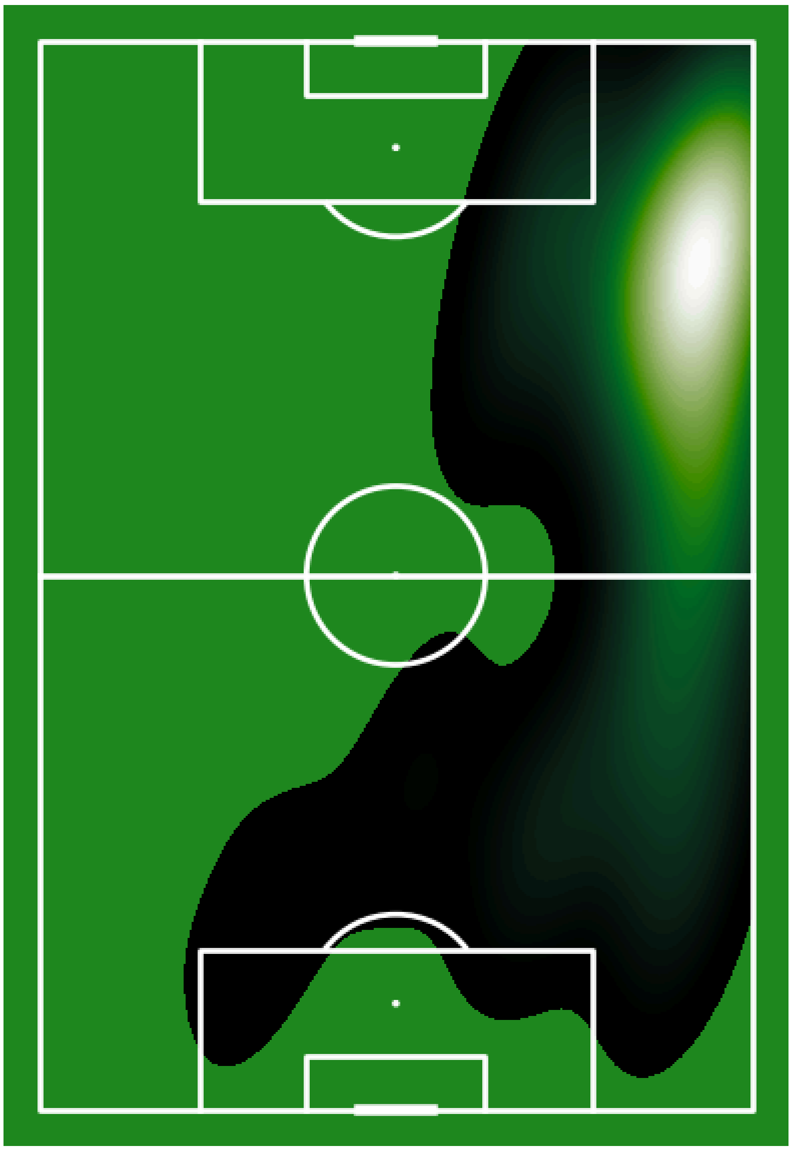
In [30]:
```
yamal_events = df_events[df_events['player_name'] == 'Lamine Yamal Nasrao
print(yamal_events.shape)
```

```
(148, 75)
```

In [31]:
```
pitch_light = VerticalPitch(line_color='white', line_zorder=2,
                            pitch_color='#228B22')
fig, ax = pitch_light.draw(figsize=(10, 6))
kdeplot_light = pitch_light.kdeplot(yamal_events.x, yamal_events.y,
                                    ax=ax, cmap=cmr.jungle,
                                    fill=True, levels=100)
```



In [32]:
```
saka_events = df_events[df_events['player_name'] == 'Bukayo Saka']

pitch_light2 = VerticalPitch(line_color='white', line_zorder=2,
                             pitch_color='#228B22')
fig2, ax2 = pitch_light2.draw(figsize=(10, 6))
kdeplot_light2 = pitch_light2.kdeplot(saka_events.x,
                                      saka_events.y, ax=ax2, cmap=cmr.jun
                                      fill=True, levels=100)
```



Pass network figure for both teams

```
In [45]: events, related, freeze, players = parser.event(3943043)
         TEAM = 'Spain'
         OPPONENT = 'Versus England 2024 Euros Final'
```

```
In [46]: events.loc[events.tactics_formation.notnull(), 'tactics_id'] = events.loc
             events.tactics_formation.notnull(), 'id']
         events[['tactics_id', 'tactics_formation']] = events.groupby('team_name')
             'tactics_id', 'tactics_formation']].ffill()
```

```
In [47]: formation_dict = {1: 'GK', 2: 'RB', 3: 'RCB', 4: 'CB', 5: 'LCB',
                           6: 'LB', 7: 'RWB', 8: 'LWB', 9: 'RDM', 10: 'CDM',
                           11: 'LDM', 12: 'RM', 13: 'RCM', 14: 'CM', 15: 'LCM',
                           16: 'LM', 17: 'RW', 18: 'RAM', 19: 'CAM', 20: 'LAM',
                           21: 'LW', 22: 'RCF', 23: 'ST', 24: 'LCF', 25: 'SS'}
         players['position_abbreviation'] = players.position_id.map(formation_dict
```

```
In [48]: sub = events.loc[events.type_name == 'Substitution',
                          ['tactics_id', 'player_id', 'substitution_replacement_id
                           'substitution_replacement_name']]
         players_sub = players.merge(sub.rename({'tactics_id': 'id'}, axis='column
                                     on=['id', 'player_id'], how='inner', validate
         players_sub = (players_sub[['id', 'substitution_replacement_id', 'positio
                        .rename({'substitution_replacement_id': 'player_id'}, axis
         players = pd.concat([players, players_sub])
         players.rename({'id': 'tactics_id'}, axis='columns', inplace=True)
         players = players[['tactics_id', 'player_id', 'position_abbreviation']]
```

In [49]:

```
# add on the position the player was playing in the formation to the even
events = events.merge(players, on=['tactics_id', 'player_id'], how='left'
# add on the position the receipient was playing in the formation to the
events = events.merge(players.rename({'player_id': 'pass_recipient_id'},
                                     axis='columns'), on=['tactics_id', '
                      how='left', validate='m:1', suffixes=['', '_receipt
```

In [50]:

```
events.groupby('team_name').tactics_formation.unique()
```

Out[50]:

```
team_name
England    [4231, 41212]
Spain             [4231]
Name: tactics_formation, dtype: object
```

In [51]:

```
FORMATION = '4231'
pass_cols = ['id', 'position_abbreviation', 'position_abbreviation_receip
passes_formation = events.loc[(events.team_name == TEAM) & (events.type_n
                              (events.tactics_formation == FORMATION) &
                              (events.position_abbreviation_receipt.notnu
                              pass_cols].copy()
location_cols = ['position_abbreviation', 'x', 'y']
location_formation = events.loc[(events.team_name == TEAM) &
                                (events.type_name.isin(['Pass', 'Ball Rec
                                (events.tactics_formation == FORMATION),
                                location_cols].copy()

# average locations
average_locs_and_count = (location_formation.groupby('position_abbreviati
                          .agg({'x': ['mean'], 'y': ['mean', 'count']}))
average_locs_and_count.columns = ['x', 'y', 'count']

# calculate the number of passes between each position
passes_formation['pos_max'] = (passes_formation[['position_abbreviation',
                                                 'position_abbreviation_re
                               .max(axis='columns'))
passes_formation['pos_min'] = (passes_formation[['position_abbreviation',
                                                 'position_abbreviation_re
                               .min(axis='columns'))
passes_between = passes_formation.groupby(['pos_min', 'pos_max']).id.coun
passes_between.rename({'id': 'pass_count'}, axis='columns', inplace=True)

# add on the location of each player so we have the start and end positio
passes_between = passes_between.merge(average_locs_and_count, left_on='po
                                      right_index=True)
passes_between = passes_between.merge(average_locs_and_count, left_on='po
                                      right_index=True,
                                      suffixes=['', '_end'])
```

In [52]:

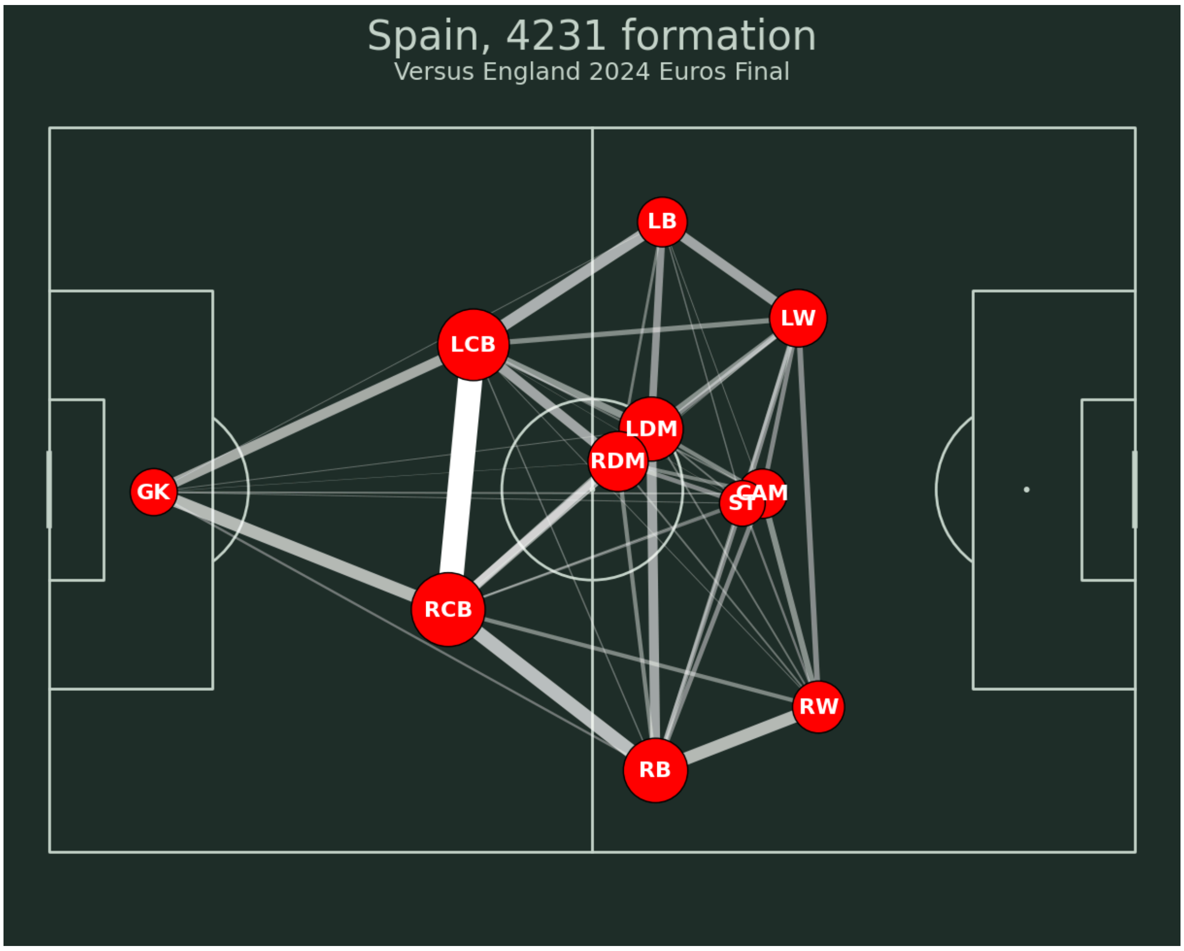
```
MAX_LINE_WIDTH = 18
MAX_MARKER_SIZE = 3000
passes_between['width'] = (passes_between.pass_count / passes_between.pas
                           MAX_LINE_WIDTH)
```

```
average_locs_and_count['marker_size'] = (average_locs_and_count['count']
                                         / average_locs_and_count['count'
                                         MAX_MARKER_SIZE)
```

In [53]:
```
MIN_TRANSPARENCY = 0.3
color = np.array(to_rgba('white'))
color = np.tile(color, (len(passes_between), 1))
c_transparency = passes_between.pass_count / passes_between.pass_count.ma
c_transparency = (c_transparency * (1 - MIN_TRANSPARENCY)) + MIN_TRANSPAR
color[:, 3] = c_transparency
```

In [58]:
```
from mplsoccer import Pitch
```

In [59]:
```
fig, axs = pitch.grid(figheight=10, title_height=0.08, endnote_space=0,
                      axis=False,
                      title_space=0, grid_height=0.82, endnote_height=0.0
fig.set_facecolor("#22312b")
pass_lines = pitch.lines(passes_between.x, passes_between.y,
                         passes_between.x_end, passes_between.y_end,
                         lw=passes_between.width,
                         color=color, zorder=1, ax=axs['pitch'])
pass_nodes = pitch.scatter(average_locs_and_count.x, average_locs_and_cou
                           s=average_locs_and_count.marker_size,
                           color='red', edgecolors='black', linewidth=1,
                           alpha=1, ax=axs['pitch'])
for index, row in average_locs_and_count.iterrows():
    pitch.annotate(row.name, xy=(row.x, row.y), c='white', va='center',
                   ha='center', size=16, weight='bold', ax=axs['pitch'])

# endnote /title
TITLE_TEXT = f'{TEAM}, {FORMATION} formation'
axs['title'].text(0.5, 0.7, TITLE_TEXT, color='#c7d5cc',
                  va='center', ha='center', fontsize=30)
axs['title'].text(0.5, 0.25, OPPONENT, color='#c7d5cc',
                  va='center', ha='center', fontsize=18)
for txt in ax.texts[:]:
    txt.remove()
```



In [60]:
```
TEAM2 = 'England'
OPPONENT2 = 'Versus Spain 2024 Euros Final'
```

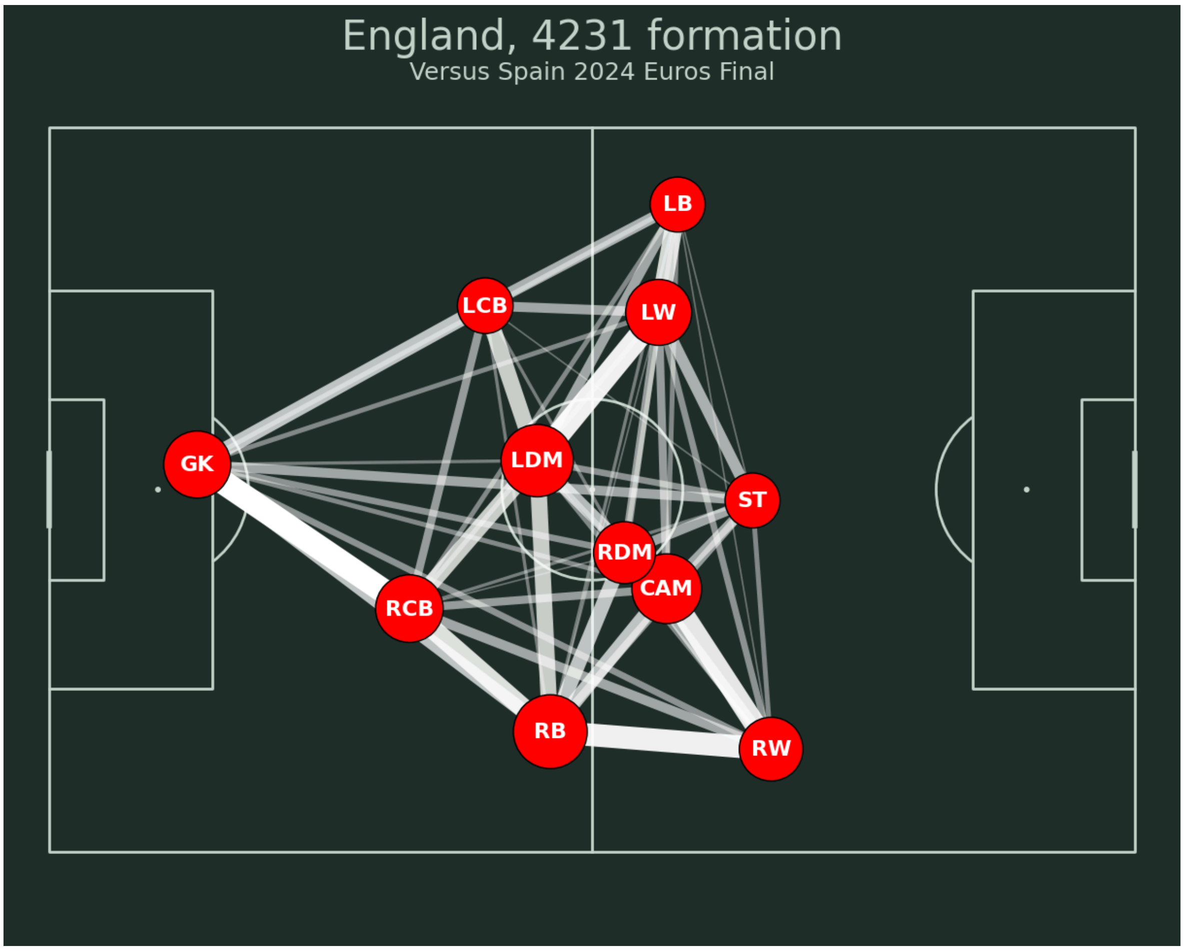
In [61]:
```
FORMATION = '4231'
pass_cols = ['id', 'position_abbreviation', 'position_abbreviation_receip
passes_formation = events.loc[(events.team_name == TEAM2) & (events.type_
                              (events.tactics_formation == FORMATION) &
                              (events.position_abbreviation_receipt.notnu
                                pass_cols].copy()
location_cols = ['position_abbreviation', 'x', 'y']
location_formation = events.loc[(events.team_name == TEAM2) &
                                (events.type_name.isin(['Pass', 'Ball Rec
                                (events.tactics_formation == FORMATION),
                                location_cols].copy()

# average locations
average_locs_and_count = (location_formation.groupby('position_abbreviati
                          .agg({'x': ['mean'], 'y': ['mean', 'count']}))
average_locs_and_count.columns = ['x', 'y', 'count']

# calculate the number of passes between each position
#(using min/ max so we get passes both ways)
passes_formation['pos_max'] = (passes_formation[['position_abbreviation',
                                                  'position_abbreviation_re
                               .max(axis='columns'))
passes_formation['pos_min'] = (passes_formation[['position_abbreviation',
                                                  'position_abbreviation_re
                               .min(axis='columns'))
passes_between = passes_formation.groupby(['pos_min', 'pos_max']).id.coun
passes_between.rename({'id': 'pass_count'}, axis='columns', inplace=True)
```

```
# add on the location of each player so we have the start and end positio
passes_between = passes_between.merge(average_locs_and_count,
                                      left_on='pos_min', right_index=True
passes_between = passes_between.merge(average_locs_and_count, left_on='po
                                      right_index=True,
                                      suffixes=['', '_end'])
```

In [62]:
```
MAX_LINE_WIDTH = 18
MAX_MARKER_SIZE = 3000
passes_between['width'] = (passes_between.pass_count / passes_between.pas
                           MAX_LINE_WIDTH)
average_locs_and_count['marker_size'] = (average_locs_and_count['count']
                                         / average_locs_and_count['count'
                                         MAX_MARKER_SIZE)
```

In [63]:
```
MIN_TRANSPARENCY = 0.3
color = np.array(to_rgba('white'))
color = np.tile(color, (len(passes_between), 1))
c_transparency = passes_between.pass_count / passes_between.pass_count.ma
c_transparency = (c_transparency * (1 - MIN_TRANSPARENCY)) + MIN_TRANSPAR
color[:, 3] = c_transparency
```

In [64]:
```
fig, axs = pitch.grid(figheight=10, title_height=0.08, endnote_space=0,
                      axis=False,
                      title_space=0, grid_height=0.82, endnote_height=0.0
fig.set_facecolor("#22312b")
pass_lines = pitch.lines(passes_between.x, passes_between.y,
                         passes_between.x_end, passes_between.y_end,
                         lw=passes_between.width,
                         color=color, zorder=1, ax=axs['pitch'])
pass_nodes = pitch.scatter(average_locs_and_count.x, average_locs_and_cou
                           s=average_locs_and_count.marker_size,
                           color='red', edgecolors='black', linewidth=1,
                           alpha=1, ax=axs['pitch'])
for index, row in average_locs_and_count.iterrows():
    pitch.annotate(row.name, xy=(row.x, row.y), c='white', va='center',
                   ha='center', size=16, weight='bold', ax=axs['pitch'])

# endnote /title
TITLE_TEXT = f'{TEAM2}, {FORMATION} formation'
axs['title'].text(0.5, 0.7, TITLE_TEXT, color='#c7d5cc',
                  va='center', ha='center', fontsize=30)
axs['title'].text(0.5, 0.25, OPPONENT2, color='#c7d5cc',
                  va='center', ha='center', fontsize=18)
for txt in ax.texts[:]:
    txt.remove()
```



Shots and Goals for Both Teams

In [65]:
```
import numpy as np
from matplotlib import colormaps
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

from mplsoccer import (VerticalPitch, Pitch, create_transparent_cmap,
                       FontManager, arrowhead_marker, Sbopen)
```

In [70]:
```
from mplsoccer import VerticalPitch
```

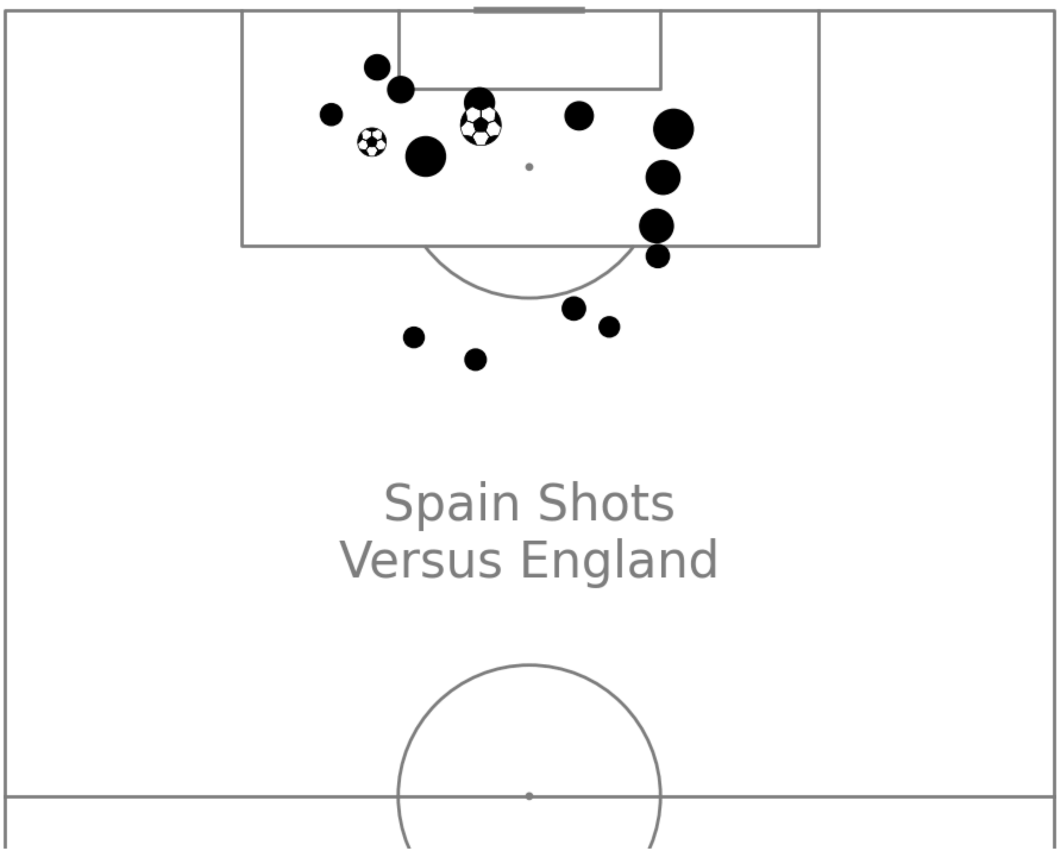
In [71]:
```
df, related, freeze, tactics = parser.event(3943043)
```

In [72]:
```
df_shots_Spain = df[(df.type_name == 'Shot') & (df.team_name == 'Spain')]
```

In [73]:
```
df_pass_Spain = df[(df.type_name == 'Pass') &
                   (df.team_name == 'Spain') &
                   (~df.sub_type_name.isin(['Throw-in', 'Corner',
                                            'Free Kick', 'Kick Off']))].c
```

In [77]:
```
from mplsoccer import VerticalPitch

# filter goals/non-goal shots
df_goals_Spain = df_shots_Spain[df_shots_Spain.outcome_name == 'Goal'].co
df_non_goal_shots_Spain = df_shots_Spain[df_shots_Spain.outcome_name != '

pitch = VerticalPitch(half=True, pitch_color='white', line_color='grey')
```

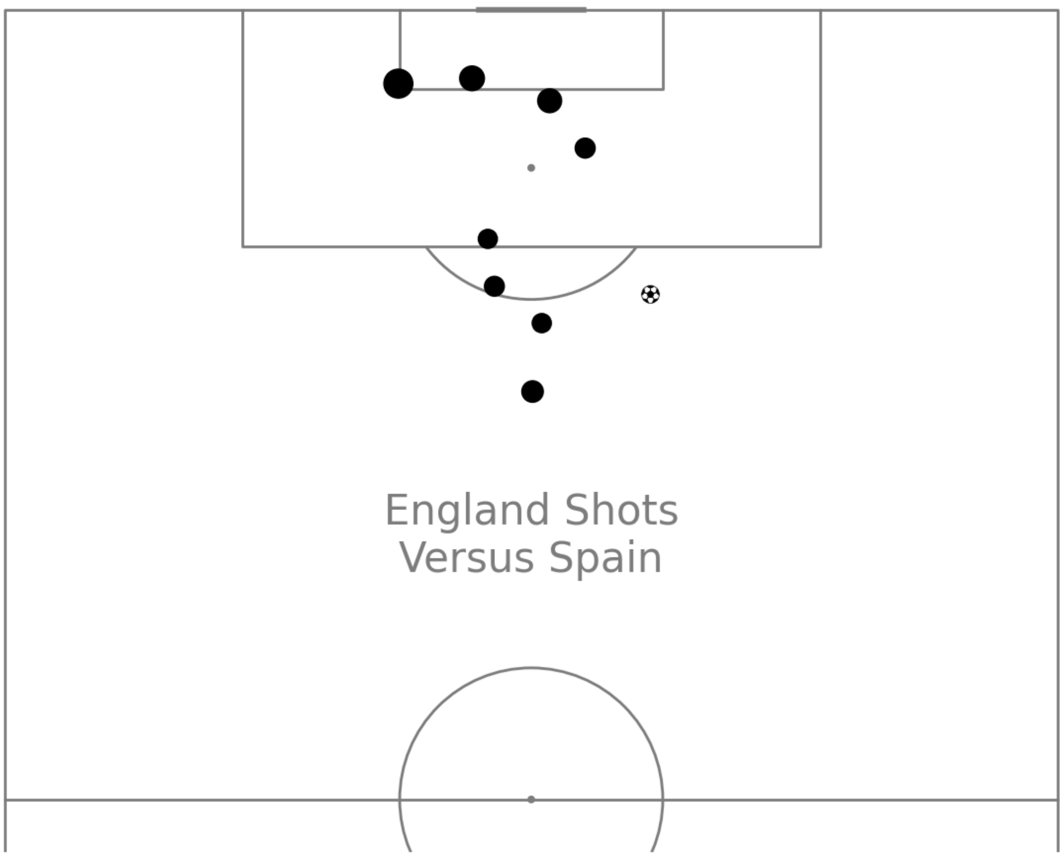
```
fig, ax = pitch.draw(figsize=(10, 8))

# plot non-goal shots with hatch
sc1 = pitch.scatter(df_non_goal_shots_Spain.x, df_non_goal_shots_Spain.y,
                    s=(df_non_goal_shots_Spain.shot_statsbomb_xg * 1900)
                    edgecolors='black',
                    c='black',
                    hatch='///',
                    marker='o',
                    ax=ax)

# plot goal shots with football marker
sc2 = pitch.scatter(df_goals_Spain.x, df_goals_Spain.y,
                    s=(df_goals_Spain.shot_statsbomb_xg * 1900) + 100,
                    edgecolors='black',
                    c='white',
                    marker='football',
                    ax=ax)

# add title text
txt = ax.text(x=40, y=80, s='Spain Shots\nVersus England',
              size=30,
              color=pitch.line_color,
              va='center', ha='center')
```



In [80]:
```
df_shots_Eng = df[(df.type_name == 'Shot') & (df.team_name == 'England')]
```

In [81]:
```
df_pass_Eng = df[(df.type_name == 'Pass') &
                 (df.team_name == 'England') &
                 (~df.sub_type_name.isin(['Throw-in', 'Corner',
                                          'Free Kick', 'Kick Off']))].c
```

In [82]:
```
# filter goals / non-shot goals
df_goals_Eng = df_shots_Eng[df_shots_Eng.outcome_name == 'Goal'].copy()
df_non_goal_shots_Eng = df_shots_Eng[df_shots_Eng.outcome_name != 'Goal']

fig, ax = pitch.draw(figsize=(12, 10))

# plot non-goal shots with hatch
sc1 = pitch.scatter(df_non_goal_shots_Eng.x, df_non_goal_shots_Eng.y,
                    s=(df_non_goal_shots_Eng.shot_statsbomb_xg * 1900) +
                    edgecolors='black',
                    c='black',
                    hatch='///',
                    marker='o',
                    ax=ax)

# plot goal shots with a
sc2 = pitch.scatter(df_goals_Eng.x, df_goals_Eng.y,
                    # size varies between 100 and 1900 (points squared)
                    s=(df_goals_Eng.shot_statsbomb_xg * 1900) + 100,
                    edgecolors='black',  # give the markers a charcoal bo
                    c='white',  # color for scatter in hex format
                    marker='football',
                    ax=ax)

txt = ax.text(x=40, y=80, s='England Shots\nVersus Spain',
              size=30,
              color=pitch.line_color,
              va='center', ha='center')
```



In [ ]: